A&A, 706, A73 (2026)
https://doi.org/10.1051/0004-6361/202556909

Astronomy & Astrophysics

Optimized smoothing kernels for smoothed particle hydrodynamics

Robert Wissing*, Thomas Quinn, Ben Keller, James Wadsley, and Sijing Shen

Institute of Theoretical Astrophysics, University of Oslo, Postboks 1029, 0315 Oslo, Norway

Received 19 August 2025 / Accepted 1 December 2025

ABSTRACT

We present a set of new smoothing kernels for smoothed particle hydrodynamics (SPH) that improve the convergence of the method without any additional computational cost. These kernels are generated through a linear combination of other SPH kernels combined with an optimization strategy to minimize the error in the Gresho-Chan vortex test case. To facilitate the different choices in gradient operators for SPH in the literature, we performed this optimization for both geometric density average force SPH (GDSPH) and linear-corrected gradient SPH (ISPH). In addition to the Gresho-Chan vortex, we also performed simulations of the hydrostatic glass, Kelvin-Helmholtz instability, Sod shocktube case, and Evrard collapse as well as a subsonic blob test. At low neighbor numbers (<128), there is a significant improvement across the different tests, with the greatest impact shown for GDSPH. Apart from the popular Wendland kernels, we also explored other positive-definite kernels in this paper, which include the "missing" Wendland kernels, Wu kernels, and the Buhmann kernel. In addition, we also present a method for producing arbitrary non-biased initial conditions in SPH. This method uses the SPH momentum equation together with an artificial pressure combined with a global and local relaxation stage to minimize local and global errors.

Key words. hydrodynamics – methods: numerical

1. Introduction

Smoothed particle hydrodynamics (SPH) is a Lagrangian particle method that has been applied to a wide range of topics within astrophysics. The traditional SPH method can be derived from the principle of least action and the Euler-Lagrange equations (Monaghan 1988; Price 2012), resulting in a numerical scheme for hydrodynamics that spatially conserves linear momentum, angular momentum, entropy, and energy exactly and that leaves the error in conservation mainly dependent on the time integration scheme[1]. Many of the drawbacks of the traditional SPH method, such as handling shocks, shear flow, convergence issues, and density gradients, have been greatly improved over the years. In addition, there are still numerous aspects of the method that remain unexplored, with the potential for even more improvements in the future. In this article we focus on improving the convergence qualities of SPH.

Since SPH is a particle-based method, the fluid is discretized by mass elements rather than volume elements, as in grid-based codes. A smoothing kernel is used to interpolate the fluid quantities at any point in space. This includes the density, which determines the effective volume of the particles. While the sum of volume elements in a grid code remains constant, the sum of particle volumes in an SPH code is highly dependent on the accuracy of the interpolation. This accuracy is mainly determined by the smoothing kernel and how well the particles are distributed within the kernel. An optimally distributed set of particles satisfies, for each particle a, the following constraints:

$$Q_{0,a} = \sum_b \frac{m_b}{\rho_b} W_{ab} = 1 \tag{1}$$

$$Q_{1,a} = \sum_b \frac{m_b}{\rho_b} (\mathbf{r}_b - \mathbf{r}_a) W_{ab} = 0, \tag{2}$$

where the summation runs over neighboring particles, b, with mass m_b, density ρ_b, and kernel weighting function W_{ab}. Here Q_0 represent the partition of unity, and Q_1 indicates that there is no preferred direction or bias in the distribution of particles. The larger the deviation from these values, the worse the interpolation of fluid quantities. Related to this is the effective area between particles, which is analogous to the flux area between volume elements in grid-based codes. In a grid-based code, the areas fully enclose the volume of the resolution element ($\int_s \hat{n} \mathrm{d}S = 0$), where S is the bounding surface of the resolution element and $\hat{n}$ is the unit normal vector pointing outward from the surface. However, in SPH this is not necessary true and will again depend on the particle distribution such that there can be a non-canceling of particle areas ($\int_s \hat{n} \mathrm{d}S \neq 0$). As momentum flux is made to be symmetric in SPH (such that momentum is conserved), non-canceling of particle areas leads to an automatic re-meshing mechanism in SPH. This causes particles to move to minimize this error and move toward a more optimal particle distribution[2]. This will occur even in the absence of hydrodynamic forces (constant pressure) and has thus come to be known as the zeroth order error of SPH. Particles that are optimally distributed fulfill the cancellation of areas and the following conditions:

$$E^i_{0,a} = 2 \sum_b \frac{m_b}{\rho_b} \overline{\nabla W_{ab}} = 0 \tag{3}$$

* Corresponding author: robertwi@astro.uio.no

[1] This is in regard to constant smoothing length or accounting for changes in smoothing length, h, through ∇h terms.

[2] When equal flux enters through each surface area between particles (e.g., for constant pressure), a disparity in effective area on either side results in movement toward the side with a smaller effective area. This occurs because the sum of the effective areas between particles and their normals do not cancel out.

$$E^{ij}_{1,a} = 2\sum_b \frac{m_b}{\rho_b}(\mathbf{r}_b - \mathbf{r}_a)^i \overline{\nabla^j_a W_{ab}} = \delta^{ij}. \tag{4}$$

Here E_0 is the zeroth order error[3] and E_1 is the linear gradient error. The indices $i, j \in \{x, y, z\}$ denote Cartesian vector components and δ^{ij} the Kronecker delta. Here we use the GDSPH gradient operator (Wadsley et al. 2017), which we continue to use for the rest of this paper. The momentum equation for SPH is derived from the Euler equations and given by

$$\frac{\mathrm{d}v}{\mathrm{d}t} = \frac{-\nabla P}{\rho} \rightarrow \frac{\mathrm{d}v_a}{\mathrm{d}t} = -\sum_b m_b \left(\frac{P_a + P_b}{\rho_a \rho_b}\right) \overline{\nabla W_{ab}}. \tag{5}$$

To get the error terms of this function, we can Taylor expand P_b around r_a:

$$\begin{aligned}\frac{\mathrm{d}v_i}{\mathrm{d}t} = &-\frac{2P_a}{\rho_a}\sum_b \frac{m_b}{\rho_b}\overline{\nabla^i_a W_{ab}} \\ &- \frac{2\nabla_i P_a}{\rho_a}\sum_b \frac{m_b}{\rho_b}(r_b - r_a)^j \overline{\nabla^j_a W_{ab}} + O(h^2).\end{aligned} \tag{6}$$

It can be seen that our momentum equation is of the second order if the particle fulfills the E_0 and E_1 conditions. Most of the time this is not the case, but we can develop optimizations to try and minimize these kinds of errors. These optimization often involve different smoothing kernels, numbers of neighbors, corrections to the gradients, different gradient operators, or different choices of effective particle volume.

Linear-exact gradient corrections have become popular in recent times due to the increased accuracy they offer in subsonic flows and in shearing flows. Here, a matrix inversion of the E_1 condition is performed and then applied to the gradient of the kernel to ensure that the E_1 condition is fulfilled. There are a few variations to this correction. The most straightforward is to simply invert E_1 in Eq. (4) (Bonet 1999; Price & Monaghan 2004). In the CRKSPH formulation (Frontiere et al. 2017) and the SPH method presented by Rosswog (2025), the kernel and its derivatives are reproduced to enforce both the zeroth order and the linear order; however, when deriving the pairwise forces to uphold linear momentum conservation, the zeroth order correction is effectively lost, and only the linear correction remains. There might be benefits in these reproducing kernels over other linear correction methods, but that remains to be seen. Another way of constructing linear-exact gradients for SPH was devised by García-Senz et al. (2012). In their proposal, gradients are calculated from an integral expression so that there is no need to explicitly calculate the analytic derivative of the kernel function. Similarly to Bonet (1999); Price & Monaghan (2004), a matrix inversion is then performed to linearly correct for the gradients. This method is known as the integral smoothed particle hydrodynamics (ISPH) scheme (García-Senz et al. 2012). Integral-based estimates have been readily applied in the past for second derivatives, as they are much less noisy than analytical second derivatives of the smoothing kernel (Brookshaw 1985; Monaghan 2005). Moreover, ISPH have shown great advantages over regular SPH in subsonic flows (García-Senz et al. 2012; Rosswog 2015; Valdarnini 2016). The disadvantages of these linear corrections include the additional computational cost and the discrepancy in errors between the density estimate and velocity gradients, potentially leading to entropy errors. Global angular momentum is also no longer fully conserved in these formulations, as the pairwise force is not always radially aligned between particles. This becomes a resolution-dependent error, and it has been stated to remain relatively small (Frontiere et al. 2017); however, this error is dependent on both resolution and the given particle distribution. On the other hand, this is offset by more precise local angular momentum transport due to more accurate gradients.

The interpolation kernel is the foundation of the SPH method and is ever present in determining the accuracy of a simulation, even when applying the optimizations mentioned above. For a long time, B-splines were the most popular smoothing kernel to use due to their compact support, good interpolation, and flexibility in polynomial degree. However, an issue with these kernels has always been that they are susceptible to the pairing-instability. The pairing-instability occurs when the number of neighbors exceeds a certain critical value, causing particles to clump together and consequently reducing the resolution of the simulation (Thomas & Couchman 1992; Morris 1996; Børve et al. 2004; Price 2012; Dehnen & Aly 2012). This critical value will depend on the kernel, but for the cubic kernel, it lies in the region of $N_{smooth} = 50$. Increasing the number of neighbors leads to more accurate low-order errors (E_0, E_1, Q_0, Q_1) and less particle noise. This is particularly desirable when modeling subsonic flows, where smooth velocity fields are essential[4]. This is why Wendland kernels have become very popular in recent times, as they are stable against the pairing instability at all neighbor numbers. It was shown in Dehnen & Aly (2012) that a positive Fourier transform is a necessary condition for stability against the pairing instability, though it is still unclear why negative values in the Fourier transform triggers the pairing instability. It has been hypothesized that this is a consequence of the particles trying to re-order themselves in order to minimize the total internal energy (U) given a fixed entropy. While the uniform particle distribution always represents a local minimum, if a paired particle distribution results in a lower internal energy, then the uniform particle distribution is only meta-stable. Cubic spline kernels are hypothesized to exhibit this meta-stable property, as it has an oscillating over-under estimation of density given a uniform particle distribution for varying neighbor numbers. The Wendland kernel and other kernels with positive definite Fourier transform seem to exhibit an overestimation of the density, which continuously decreases as number of neighbors increases. Dehnen & Aly (2012) hypothesized that pairing occurs when the following condition is fulfilled: $\rho(N_{smooth}/f) < \rho(N_{smooth})$; for $1 < f \leq 2$. The overestimation of the density by the Wendland kernel can be especially high when using low neighbor numbers. Nevertheless, one can correct for the kernel bias by adjusting the self-interaction term ($W(0, h)$), thus effectively removing this bias from the density estimate while leaving the gradients unaffected. Though the Wendland kernels have been popularized, there are many other interpolation kernels that meet the criteria of having a positive Fourier transform, Gaussian shape, and compact support. These include the Wu family of kernels (Wu 1995), the missing Wendland kernels (Schaback 2011), and the Buhmann family of kernels (Buhmann 1998). Another family of kernels is the Sinc kernels, $W(q) = \left(\sin(\frac{\pi}{2}q)/(\frac{\pi}{2}q)\right)^n$ (Cabezón et al. 2008; García-Senz et al. 2014), which is directly linked to the Dirac-δ function. Sinc kernels, as with the B-spline family of kernels, can be subject to the pairing instability. However,

[3] The gradient of the smoothing kernel gives the effective area between two particles.

[4] The drawbacks of using a high number of neighbors is the additional computational cost and the increase in smoothing length (h), which can lead to excessive smoothing of sharp features in the flow.

the critical neighbor number for the instability can be pushed higher by increasing the sharpness of the kernel, which is controlled by the kernel exponent n. A consequence of increasing the sharpness is that it usually gives a worse interpolation at lower neighbor numbers. This is likely related to the larger difference in weight between the inner and outer regions produced by sharper kernels. Recently, a linear combination of two Sinc kernels was proposed by Cabezón & García-Senz (2024) to generate a kernel that is more resistant to the pairing instability while retaining good interpolation properties over a wide range of neighbors numbers. In this work, we aim to design smoothing kernels specifically optimized for SPH rather than using interpolation kernels developed for other purposes. Our approach involves optimizing a linear combination of kernels, similar to the method in Cabezón & García-Senz (2024), to produce the most accurate kernel for a given neighbor number. A big challenge in this work was determining the most natural initial conditions to minimize any kind of bias.

Lattice initial conditions are often used to provide an initially optimal distribution of particles. In this case, particles are put in a cubic or closed package lattice. At first this might seem to be a nice solution, as the E_0 and E_1 conditions are fulfilled. But there are many disadvantages to actually using a lattice for the initial condition. First, the lattice structure is easily broken by shocks and shear flows, which will generate aggressive noise as the particles move off the lattice structure, and this will cascade through the simulation volume. Second, particle lattice distributions can introduce directional bias and simulation artifacts. For instance, a shock wave propagating along a preferred direction of the lattice can disproportionately gather particles in that direction, leading to undesirable oscillations. Third, the two issues above are exacerbated when dealing with density gradients in the initial conditions. Lattice particle distributions can give both an artificial positive result for SPH, as it provides very high accuracy as long as the lattice holds up, but also an artificial negative result, as significant noise is generated when the lattice breaks down. A much more natural distribution for SPH is the glass distribution, which SPH will always strive to re-mesh toward as particles are disturbed by such forces as shocks and shear flows. Glass distributions are often obtained by letting the particles relax, beginning with a lattice or random distribution, adding some random velocities, and using a velocity damping term to eventually approach a steady state. This can be done with the influence of gravity to generate collapsed density structures or without it to generate uniform distributions. However, this method has limitations and downsides. First, one cannot generate density contrasts for setups that are purely hydrodynamic[5] (the Kelvin-Helmholtz setup for example). Second, glass generation performed in this way can be costly, as oscillations can take a long time to damp. Third, damping the velocities too quickly can lead to inaccurate distributions. In this paper, we have instead developed a glass generator that relies on the SPH method to find a relaxed glass distribution for arbitrary density gradients.

The use of lattice initial conditions is especially problematic when it comes to assessing the quality of a smoothing kernel, as the results do not reflect the natural distribution of particles and give a positive bias toward smoother kernels, which usually have larger E_0 errors in glass distributions (given the same neighbor count). In Dehnen & Aly (2012), the correction for the kernel bias was made based on the bias when interpolating a closed-packed lattice distribution. The bias was also given by a simple power law. In this paper we calculate a new correction to the kernel bias for all our kernels that is based on the generated glass that each kernel itself produces. We also give a piecewise function to capture the kernel bias more accurately over a wider-range of neighbor numbers.

In Sect. 2 we present all the smoothing kernels used in this paper. In Sect. 3 we go through the methods used in this paper, including the SPH methods, the IC generator, adjustment of density bias, and the optimization strategy. In Sect. 4 we present the results from the tests. And finally in Sect. 5 we discuss our results.

2. Kernel properties

All the kernels we go through in this section go to zero at

$$q = (r/2h) > 1, \tag{7}$$

even if not stated explicitly in the equations. All kernels are normalized with

$$\sigma = \frac{1}{\int_0^1 \int_0^\pi \int_0^{2\pi} W(r) r^2 \sin(\theta) \mathrm{d}r\mathrm{d}\theta\mathrm{d}\psi}. \tag{8}$$

A useful tool for analyzing kernels is their Fourier transform. Despite having similar shapes, kernels can exhibit quite different Fourier transforms. As previously mentioned, negative values in the Fourier transform are responsible for the pairing instability. The lower the wave numbers at which these negative values appear, the lower the critical number of neighbors required to trigger this instability. The Fourier transform is defined as

$$F_3[W(r)](k) = 4\pi k^{-1} \int_0^\infty \sin(kr) W(r) r \mathrm{d}r, \tag{9}$$

where $W(r)$ is the smoothing kernel and k is the wave number. All the kernels that we mention in this section are positive-definite, which means that they have as strictly positive Fourier transform. Positive definite kernels all produce a general over-bias in their density estimation, which become more prominent at low neighbor numbers, as the bias is dominated by the self contribution. The neighbor number at which this bias becomes negligible depends on the smoothing kernel (higher order/more peaked kernels give higher bias). This self contribution can be corrected for as described in Sect. 3.3.

2.1. Generalized Wendland kernel

The Wendland kernels is a family of compactly supported radial functions generated by a dimension walk (integration) of a truncated power function (Wendland 1995).

$$\begin{aligned} \psi_l(r) &= (1-r)_+^l, \\ l(k) &= \lfloor 1.5 + k \rfloor + 1, \\ \phi_k(r) &= \mathcal{I}^k \psi_l. \end{aligned} \tag{10}$$

Here k is the smoothness parameter, $l(k)$ is the order of the kernel, and $\lfloor x \rfloor$ is the floor operator. We set $l(k)$ so that the generated function is strictly positive definite and radial on R^3 (following $D \leq 2l - 1$ for D dimensions). The Wendland kernels produce the minimal polynomial degree for a given space dimension and

[5] One can take two glass boxes with different densities and put them next to each other; however, the interface in such a setup is not in a relaxed distribution. And if one wants to smooth the density contrast, this also becomes problematic.

smoothness. While initially only determined for positive integer smoothness parameters, this property has been generalized for non-integer smoothness parameters in the generalized Wendland functions (Chernih & Hubbert 2014):

$$\phi_k(r) = \frac{\Gamma[l(k)+1]}{2^{l(k)+k}\Gamma[l(k)+k+1]}\left(1-r^2\right)^{l(k)+k} r^{-l(k)} \times {}_2F_1\left(\frac{l(k)}{2}, k+\frac{l(k)+1}{2}, l(k)+k+1, 1-\frac{1}{r^2}\right). \quad (11)$$

Here Γ is the gamma function and ${}_2F_1$ is the hyper-geometric function. The family of Wendland kernel functions for 3D is then defined as

$$W(r) = \sigma\phi_k(r). \quad (12)$$

Positive integers of k represent the classic Wendland kernels, where the resulting kernels have polynomial structure. The Wendland kernels with positive integers used in this paper ($k = 0, 1, 2, 3$) are

$$W_{C0}(q) = \sigma(1-q)^2, \quad (13)$$

$$W_{C2}(q) = \sigma\left((1-q)^4(1+4(1-q))\right), \quad (14)$$

$$W_{C4}(q) = \sigma\left((1-q)^6(1+6(1-q)+\frac{35}{3}(1-q)^2)\right), \quad (15)$$

$$W_{C6}(q) = \sigma\left((1-q)^8(1+8(1-q)+25(1-q)^2+32(1-q)^3)\right). \quad (16)$$

Positive half-integers are known as the missing Wendland kernels (Schaback 2011). They are compactly supported and polynomial but with additional logarithmic and square-root terms ($k = 0.5$):

$$W_{CM_{05}} = \sigma\left(\sqrt{1-q^2}+6.5\,q^2\sqrt{1-q^2} +1.5\,q^2(4+q^2)\left(\log q - \log\left(1+\sqrt{1-q^2}\right)\right)\right). \quad (17)$$

Positive non-integer kernels can also be generated; these involve a more complicated form ($k = 0.4$):

$$W_{CM_{04}} = \sigma\frac{0.375(1-q^2)^{2.4}\,{}_2F_1\left(1, 1.9, 3.4, 1-\frac{1}{q^2}\right)}{q^2}. \quad (18)$$

To avoid handling hyperbolic functions within our code we approximate this function with a polynomial fit. Higher-order kernels have smoother derivatives, which in combination with higher neighbor numbers act to decrease the sensitivity to particle disorder. The trade-off of using a higher-order kernel is that it becomes less accurate than its lower-order counterpart at low neighbor numbers (Monaghan 1992; Rosswog 2015). We refer to the Wendland kernels as C2, C4, C6, CM05, and CM04 throughout this paper.

2.2. Wu kernel

The Wu approach starts with the function (Wu 1995)

$$\phi = (1-r^2)_+^l. \quad (19)$$

Another function is then generated by convolution:

$$\phi_\ell(r) = \int_{-1}^{1}(1-t^2)^\ell(1-(2r-t)^2)^\ell\,\mathrm{d}t. \quad (20)$$

Using this function, the Wu family of kernels is generated by dimension walk (derivative):

$$\phi_{l,k} = \mathcal{D}^k\phi_l. \quad (21)$$

The Wu kernel ($\phi_{l,k}$) is strictly positive definite and radial in R^D for $D \leq 2k+1$, with polynomial of degree $4l-2k+1$ and smoothness of $C^{2(l-k)}$, which indicates how many times the kernel is continuously differentiable ($2(l-k)$ times). The polynomial degree is higher for the Wu kernel than the Wendland kernel at a certain smoothness. In this paper we have chosen to look at the $\phi_{1,2}$ and $\phi_{1,3}$:

$$W_{WU2} = (1-q)_+^4(4+16q+12q^2+3q^3), \quad (22)$$

$$W_{WU4} = (1-q)_+^6(6+36q+82q^2+72q^3+30q^4+5q^5). \quad (23)$$

For a prescribed smoothness the polynomial degree of Wendland kernels are lower than that of Wu kernels. An interesting thing about Wu kernels is that their Fourier transform is positive, decays quickly, and goes to zero in periodic intervals. We refer to the Wu kernels as WU2 and WU4 throughout this paper.

2.3. Buhmann kernel

Buhmann's approach (Buhmann 1998) involves integrating a positive function $f(t) = t^\alpha(1-t^\delta)_+^\rho$ against a strictly positive definite kernel $K(t,r) = \left(1-\frac{r^2}{t}\right)_+^\lambda$. Buhmann's family of kernels thus take the general form

$$\phi(r) = \int_0^\infty (1-r^2/t)_+^\lambda t^\alpha(1-t^\delta)_+^\rho \mathrm{d}t. \quad (24)$$

Here $0 < \delta \leq 0.5$, $\rho > 0.5$ and the kernel is strictly positive and radial in $R^D \leq 3$ for $\lambda \geq 0$ and $-1 < \alpha < \frac{\lambda-1}{2}$. Buhmann's family of kernels consists of a polynomial and an additional logarithmic term. The Buhmann kernel that we include in this paper is ($\alpha = \delta = \frac{1}{2}$ $\rho = 1$ $\lambda = 2$ from Buhmann 2003):

$$W_{Buh}(r) = \sigma\left(12q^4\log q - 21q^4 + 32q^3 - 12q^2 + 1\right). \quad (25)$$

We refer to this kernel as BUH throughout this paper.

3. Method

3.1. SPH methods

We performed the optimization using both the GDSPH and ISPH methods. We did this because the "best" kernel may differ between the two methods due to the removal of the linear error term. The density of both GDSPH and ISPH was calculated using the standard density estimate

$$\rho_a = \sum_b m_b W_{ab}(h_a). \quad (26)$$

We used the same artificial viscosity scheme for all the simulations. To keep things as simple as possible, we implemented a classic SPH artificial viscosity (Monaghan 1992) with constant α and β. We used $\alpha = 1$ and $\beta = 2$ for the shock cases (Sod-shocktube and Evrard collapse), and for the subsonic cases (Gresho-Chan, Kelvin-Helmholtz, and the subsonic blob test) we used $\alpha = 0.05$ and $\beta = 2$.

3.1.1. GDSPH gradient

We implemented the GDSPH gradient operators of Gasoline2 (Wadsley et al. 2017) in this paper. This variant of SPH is identical to regular SPH when the density is uniform, but it improves gradient accuracy in cases involving sharp density gradients. The momentum equation and internal energy equation in GDSPH is written as:

$$\frac{\mathrm{d}\mathbf{v}_a}{\mathrm{d}t} = -\sum_b m_b \left[\frac{f_a P_a + f_b P_b}{\rho_a \rho_b} \overline{\nabla_a W_{ab}} \right], \tag{27}$$

$$\frac{\mathrm{d}u_a}{\mathrm{d}t} = \frac{f_a P_a}{\rho_a} \sum_b \frac{m_b}{\rho_b} \mathbf{v}_{ab} \cdot \overline{\nabla_a W_{ab}}. \tag{28}$$

Here, f_a are entropy correction terms to remove the linear entropy error caused by a mismatch between GDSPH operators and the density estimate (see Wadsley et al. 2017).

3.1.2. Integral based gradient

In ISPH (García-Senz et al. 2012), the gradients are calculated from an integral expression so that there is no need to explicitly calculate the analytic derivative of the kernel. The kernel gradients are also linear-corrected by the use of a matrix inversion of the E_1 error. Effectively, we replace the kernel gradients of Eqs. (27) and (28) with ($\overline{\nabla_a W_{ab}} \rightarrow \overline{G^k_{ab}}$):

$$(G_{a,j})^k = \sum_{d=1}^{3} C_a^{kd} r_{ba}^d W_{ab}(h_j), \tag{29}$$

$$(G_{b,j})^k = \sum_{d=1}^{3} C_b^{kd} r_{ba}^d W_{ab}(h_j), \tag{30}$$

where j is either a or b.

$$\overline{G^k_a} = 0.5(G^k_{a,a} + G^k_{a,b}), \tag{31}$$

$$\overline{G^k_b} = 0.5(G^k_{b,a} + G^k_{b,b}). \tag{32}$$

The linear correction matrix, C, is calculated with

$$C_a^{ki} = \left(\sum_b \frac{m_b}{\rho_b} r^k_{ba} r^i_{ba} \overline{\nabla_a W_{ab}} \right)^{-1}. \tag{33}$$

This form is slightly different than the one proposed by previous authors(García-Senz et al. 2012; Rosswog 2020). This is because it involves the symmetric kernel $\overline{\nabla_a W_{ab}}$ within the linear correction matrix, instead of the one-sided kernel gradient $\nabla_a W_{ab}(h_a)$. The momentum and internal energy equations become

$$\frac{\mathrm{d}\mathbf{v}_a}{\mathrm{d}t} = -\sum_b m_b \left[\frac{P_a \overline{G^k_a} + P_b \overline{G^k_b}}{\rho_a \rho_b} \right], \tag{34}$$

$$\frac{\mathrm{d}u_a}{\mathrm{d}t} = \frac{P_a}{\rho_a} \sum_b \frac{m_b}{\rho_b} \mathbf{v}_{ab} \cdot \overline{G^k_a}. \tag{35}$$

This matrix inversion requires us to do an extra neighbor loop and requires us to save six additional quantities per particle (symmetric matrix). This linear correction can introduce a global angular momentum error as pairwise torques can be generated between particles. This is because the forces between particles are no longer ensured to be aligned along the line connecting particles. This is in general true for any kernel correction of order greater than zero. This effectively results in a resolution/kernel dependent error in global angular momentum conservation.

3.2. Initial condition: Glass generator

As discussed in the introduction, producing natural and non-biased initial conditions are important for SPH. Of particular difficulty is the generation of non-uniform density distributions. Lattice initial conditions can be used in non-uniform density distributions, by either varying particle mass (Lombardi et al. 1999) or stretching a uniform lattice (Rosswog 2009; Price et al. 2018). The main issue with these methods are the use of the lattice initial condition, which as discussed are artificial, biased and experience excessive noise when the lattice breaks. Varying particle masses are undesirable as well, due to particles having more difficulty in balancing the particle distribution and generally leading to larger gradient errors and noise. In Diehl et al. (2015) a more general initial condition generator is proposed, using the concept of weighted Voronoi tessellation. This works well in generating non-uniform density distributions. The main issue with this approach is that the generated distribution does not necessarily correspond to the same natural particle distribution that a given SPH method would produce, as it does not take into account the smoothing kernel or the gradient operators. This means that it will also represent a slightly biased initial condition. In Rosswog (2020), a more SPH-like version of this method is applied, where the SPH momentum equation is used, together with an artificial pressure based on the current density error. This is an excellent approach for generating non-biased initial conditions for SPH, as it ensures a near noise free distribution at the start of your simulation (as one is using the same momentum equation to evolve the system). We propose a similar method in this paper, but add some important alterations to ensure that both local and global errors are minimized. In addition, this method will relax to a given density profile, no matter what your starting initial condition is (given that the mass of the system is correct). Similar to Rosswog (2020), we used an iterative method where during each iteration, i, we updated the position directly for each particle:

$$\mathbf{r}_{a,i+1} = \mathbf{r}_{a,i} + \Delta\mathbf{r}_{a,i} \tag{36}$$

The displacement of each particle is determined by the following equations:

$$\Delta r_{a,i} = K_{global,i} \frac{\mathrm{d}v_a/\mathrm{d}t}{f_{max,i}} h_a \tag{37}$$

$$f_{max,i+1} = \max\left(\left| \frac{\mathrm{d}v}{\mathrm{d}} \right| \right). \tag{38}$$

The term $K_{global,i}$ will determine the largest multiple of the smoothing length that a particle can be displaced. The variables $\frac{\mathrm{d}v}{\mathrm{d}t}$ and ρ_i are calculated with the chosen density estimate and gradient operator (in this paper GDSPH (Eq. (27)) and ISPH

(Eq. (34))). Hence the produced glass will represent the relaxed distribution of the chosen gradient operator, smoothing kernel and neighbor number. The pressure in the momentum equation is replaced by an artificial pressure given by

$$P_a = \left(\frac{\rho_{tar}}{\rho_a}\right)^n. \tag{39}$$

Here ρ_{tar} is the target density at the current position and ρ_a is the estimated density for the particle (Eq. (26)). Higher n means that fitting to the density is more important than balancing the particle distribution, that is, reducing E_0. This is because a minimization in density error does not necessary mean a minimization in E_0. As such, density errors and particle distribution errors will drive the particles into a position that minimizes these errors, just as the case for SPH naturally[6]. The term $K_{global,i}$ is adjusted after each step following the instructions outlined below:

- If the step leads to an overall worse mean $\sqrt{\sum\left(\frac{\mathrm{d}v_i}{\mathrm{d}t}\right)^2}$, then decrease $K_{global,i+1} = \frac{1}{d_1}K_{global,i}$.
- If the step leads to an overall better mean $\sqrt{\sum\left(\frac{\mathrm{d}v_i}{\mathrm{d}t}\right)^2}$, then increase $K_{global,i+1} = d_2 K_{global,i}$.
- Relaxation requires that $d_1 > d_2 > 1.0$. We set $d_1 = 1.8$ and $d_2 = 1.1$ during the global relaxation and $d_1 = 1.1$ and $d_2 = 1.002$ during local relaxation.
- The global relaxation stage quickly reduces the step size to interparticle size to get a good gradient estimate of the current global density profile. The step size is then increased again to a factor of the initial ($C_{loop,n}K_{global,0}$). Here we set $C_{loop,0}$ to 0.5 and reduce it by half each time $C_{loop,n} = 0.5C_{loop,0}$ if there is no improvement to the global density error. If one starts from an IC that captures the global IC well, then this step is not necessary, and only local relaxation is needed.
- In the local relaxation stage, which occurs when global relaxation stage is finished, the $K_{global,i}$ in Eq. (37) is near the average interparticle scale. From here on we slow down the rate of relaxation to ensure a more accurate local particle distribution. We stop the IC generator when $\max(\Delta r_a/h_a) < 2 \cdot 10^{-3}$, at which point particle motions are negligible, and we consider the distribution relaxed.

The initial value of $K_{global,i}$ is a parameter that can be chosen, but by default it is set to half the system size, as this ensures that the desired density profile is obtained regardless of the input IC. A much lower value can be used if one has an IC that is close to the desired density profile.

Results from the IC generator in the case of the $\Delta\rho = 8$, $n_x = 64$, $N_{smooth} = 64$ Kelvin-Helmholtz IC of Sect. 4.3 can be seen in Fig. 1. From this we can see that we have a proper glass structure in the particle distribution that follows the analytical density profile very closely. As different relaxed glasses will be produced by different smoothing kernels and neighbor numbers, we run the relaxation process for each new kernel and neighbor number when generating ICs in this paper.

3.3. Adjusting density bias

Before we optimized our linear combined kernel, we adjusted the self-bias of all our kernels. This mainly effects the density,

[6] One could even set $n = 0$, that is $P_a = 1$ and relax the distribution, this would disregard the density estimate and instead assume an exact value of the density. This could be used if one evolves the density from the continuity equation $\frac{\mathrm{d}\rho}{\mathrm{d}t}$ for example, to minimize gradient errors.

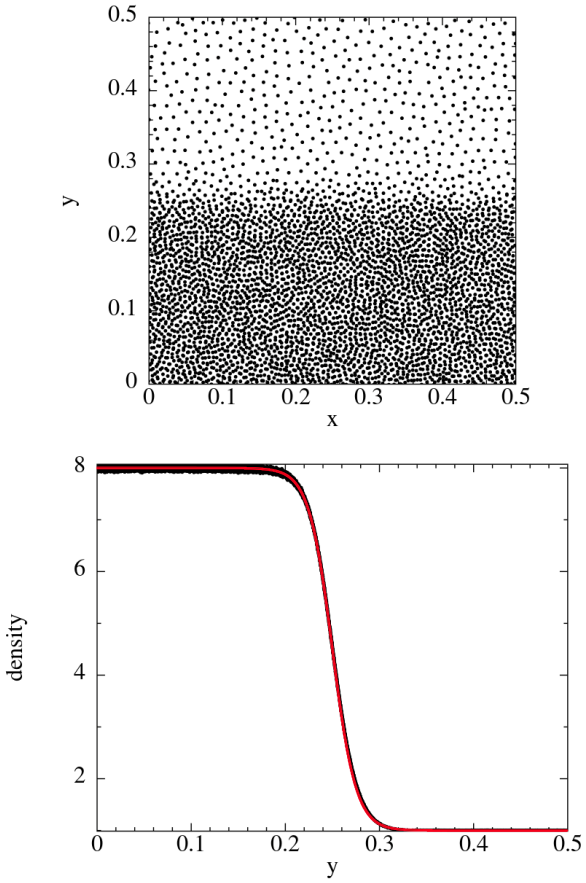

Fig. 1. Relaxed glass IC generated by the IC generator outlined in Sect. 3.2. Here we model the IC of the Kelvin Helmholtz instability (Sect. 4.3), with a density contrast of $\Delta\rho = 8$. *Top*: particle distribution within a thin slice of thickness $\Delta z = 0.015$. *Bottom*: density profile in the y direction. The red line depicts the analytical density profile and the black dots the density of the particles. The glass generated density profile is in excellent agreement with the analytical solution.

giving us a corrected density estimate:

$$\rho_a = \rho_{a,estimate} - \epsilon m_a W(0, h_a). \tag{40}$$

This density correction is a constant and is applied uniformly to all particles. We adjust ϵ so that the density of a relaxed hydrostatic glass is as close as possible to $\rho = 1.0$ (given that $M_{tot}/V_{tot} = \rho = 1.0$) for a wide range of neighbor numbers. A uniform glass produced by the IC generator will locally be independent of the resolution and only be a function of the number of neighbors and the given kernel. As such we can run low resolution cases with our IC generator and adjust the kernel bias until we produce a glass with a mean density error of around $\rho_{err} \approx 10^{-5}$, which is much less than the general SPH density noise. We run this for a wide range of neighbor numbers ($N_{smooth} \in \{16, 24, 32, 48, 64, 96, 128, 256, 512\}$). After 512 neighbors, the kernel bias is set to 1. We could then interpolate between ranges using a linear piecewise function to correct the kernel:

$$\epsilon(N_{smooth}) = \frac{c_{i+1} - c_i}{N_{min,i+1} - N_{min,i}}(N_{smooth} - N_{min,i}) + c_i. \tag{41}$$

Here, c_i is the ϵ value at $N_{smooth} = N_{min,i}$. This was performed for both GDSPH and ISPH. The coefficients and neighbor number can be found in Table B.1 for GDSPH and Table B.2 for ISPH.

3.4. Optimization strategy

In this method, we generate a new kernel by linear combination:

$$W_{new,i}(\alpha_i, \alpha_{i+1}) = (\alpha_i W_{new,i-1} + (1-\alpha_i)W_i)\,\alpha_{i+1} + (1-\alpha_{i+1})W_{i+1}. \quad (42)$$

Here, α_i and α_{i+1} are the coefficients that we optimized for. The process starts with $i = 1$ and an initial kernel $W_{new,0}$. After determining α_i, α_{i+1} by minimizing some cost function, we generated a new kernel ($W_{new,i}$). We then repeated this process with the newly generated kernel and two other kernels. This iterative process continued until we used all the kernels that we wished to include in our linear combination. We used the differential evolution method to find the global minimum of the two parameters for each iteration. We set the initial parameter value to $(\alpha_i, \alpha_{i+1}) = [0.0, 0.0]$, and we set the minimum and maximum parameter boundaries to be $bnds = [(-3.0, 3.0), (-3.0, 3.0)]$.

After the optimization was completed, we fit the linear-combination of kernels to a polynomial in the form

$$P(x) = \sum_{i=0}^{8} c_i x^i \quad (43)$$

$$W(q) = \sigma\left(1 + q^2 P(q)\right). \quad (44)$$

During the work of this paper we used two different cost functions and two different cases to optimize the kernel for. These are described in the sections below.

3.4.1. Optimized for hydrostatic glass

Perhaps, one of the simplest cases to determine the quality of a smoothing kernel comes from it's ability to generate a hydrostatic glass that minimizes the leading errors of SPH. The hydrostatic glass models a homogeneous system in pressure equilibrium. A cubic box with length $L = 1$ and total mass $M_{box} = 1$ is setup. Particles are randomly distributed throughout the box. Due to the inherent re-meshing property of SPH, the particles will be reorganized to a glass-like structure by the low order errors (Eqs. (3) and (4)). At some point the distribution will be organized into a relaxed glass, where the errors have been minimized (there is minimal/oscillatory change in the mean errors from this point onward). The distribution and the magnitude of the errors in this final relaxed state will depend on both the smoothing kernel, number of neighbors and the SPH method. The leading order error (E_0) will locally be independent of the resolution (for uniform density), so we use a relatively low resolution of $N = 32^3$ particles. We hypothesized that the relaxed glass with the smallest errors/noise would provide the best result in dynamic cases as well. We defined our cost function for this case as

$$F = \delta\rho_{err} + E_0 + 0.25E_1. \quad (45)$$

Here E_0 and E_1 are given by the norm of the error vectors defined in (Eqs. (3) and (4)) respectively. The term $\delta\rho_{err}$ is given by

$$\delta\rho_{err} = \frac{1}{N}\sum_i |\rho_i - \rho_{mean}|. \quad (46)$$

Each kernel has had its self-bias adjusted, so that ρ_{mean} will be within 10^{-5} of $\rho = 1.0$, for their respective glass and kernel (see Sect. 3.3), such that we accurately measure the density noise. We only optimized this case with the GDSPH formula, and we refer to these kernels as $O_{glass,N_{smooth}}$ ($N_{smooth} = 32, 64, 128, 256$) for the rest of the paper. The resulting polynomial coefficients for the optimization are given in Table C.1.

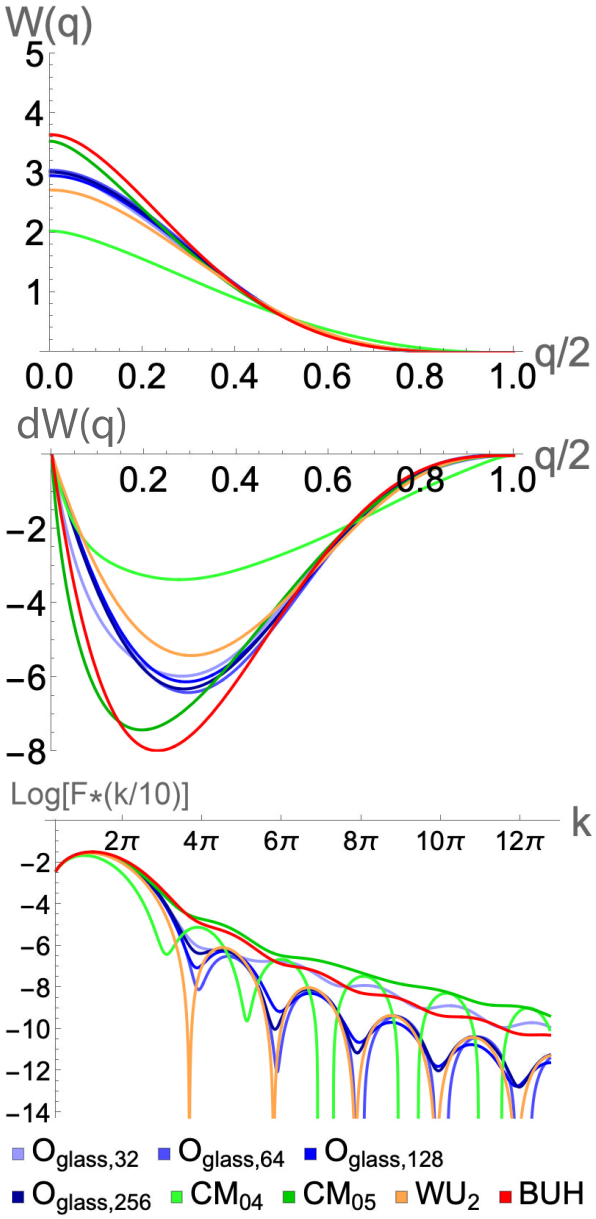

Fig. 2. *Top*: optimized kernel functions for a glass distribution (Sect. 3.4.1) together with the missing Wendland kernels (CM_{04}, CM_{05}), the Buhmann kernel (BUH), and the Wu kernel (WU_2). *Middle*: corresponding derivative of these kernel function. *Bottom*: fourier transform of these kernels, scaled by $k/10$. The optimized kernels are quite similar, with some small variations being seen in the Fourier transform. The $O_{glass,32}$ kernel deviates the most from the other kernel, with less deep troughs in the Fourier transform. These kernel's generate excellent glass distributions but degrade significantly in dynamical simulations.

The kernel function, its derivative and its Fourier transform can be seen in Fig. 2 for the optimized hydrostatic glass kernels together with WU_2, BUH, CM_{04}, CM_{05} kernels. All the optimized kernels for different N_{smooth} look very similar to each other, and they are a bit steeper than the WU_2 kernels. However, some differences can be seen in the Fourier transform, where

$O_{glass,32}$ has quite a flat trend (similar to BUH kernel) , the higher N_{smooth} kernels experience stronger oscillations (behavior somewhere in-between the WU_2 kernel and the BUH kernel). Even though we allow for the linear combination to result in negative Fourier transforms, we only end up with one kernel that has negative Fourier transform in the high wave number regime ($O_{glass,64}$). The performance of these kernels are discussed more in the next section.

3.4.2. Optimized for Gresho-Chan vortex

Initially, we assumed that a smoothing kernel with minimal errors in its final relaxed glass state would also yield the best results in dynamic simulations, but this turned out to not be the case. As different kernels have different sensitivity when it's glass gets disturbed. This was clearly seen when testing out the Gresho-Chan vortex test case. Here, even though the density is uniform, the particle distribution is continuously disturbed by a shear flow.

The Gresho-Chan vortex simulates an inviscid fluid vortex in force equilibrium (Gresho & Chan 1990). The setup is as follows: We setup a 3D thin periodic box with length $L = (1, 1, 2\frac{\sqrt{6}}{n_x})$, here n_x is the average number of particles in the x direction, and the z boundary is set to be roughly 24 particle spacings. The pressure profile is setup following

$$P(r) = \begin{cases} 5 + \frac{25}{2}r^2 & \text{for } 0 \leq r < 0.2 \\ 9 + \frac{25}{2}r^2 - 20r + 4\ln 5r & \text{for } 0.2 \leq r < 0.4 \\ 3 + 4\ln r & \text{for } r \geq 0.4 \end{cases}, \tag{47}$$

and the velocity profile follows

$$v_\theta(r) = \begin{cases} 5r & \text{for } 0 \leq r < 0.2 \\ 2 - 5r & \text{for } 0.2 \leq r < 0.4 \\ 0 & \text{for } r \geq 0.4 \end{cases}. \tag{48}$$

Even though conservation of angular momentum within SPH is near exact, the test is challenging for SPH due to the artificial viscosity induced by the particle noise and the errors in the linear gradient (Eq. (4)). The artificial viscosity will cause angular momentum transport, which will lead to mass accretion toward the center and a loss in kinetic energy. Increasing the number of neighbors (Read & Hayfield 2012; Rosswog 2015), using artificial viscosity switches (Cullen & Dehnen 2010), and doing linear gradient corrections (García-Senz et al. 2012; Valdarnini 2016; Cabezón et al. 2017) all improve the convergence of this case.

While testing the Gresho-Chan vortex we found that only $O_{glass,32}$ outperformed the other kernels for the N_{smooth} it had been optimized for, while all the other O_{glass} kernels performed rather badly (can be seen in Fig. 5). Similarly, we saw bad result for all kernels that exhibit strong oscillatory behavior in their Fourier transform, such as the WU_2 kernel (Fig. 2). Therefore we decided to additionally optimize kernels based on results from the Gresho-Chan vortex following the procedure laid out in the previous sections. The main difference is the cost function, which we set to be the $L_{1,error}$ at $t = 1$:

$$F = L_{1,error}(t = 1), \tag{49}$$

$$L_{1,error} = \frac{1}{N}\sum_{i=1}^{N}|v^a_{\phi,i}(r) - v^d_{\phi,i}(r)|. \tag{50}$$

Here $v^a_{\phi,i}$ stands for the analytical, and $v^d_{\phi,i}$ is the result from the simulation. Similarly to all other setups, we used the IC generator (Sect. 3.2) to generate an initial glass for each kernel and neighbor number before each run. We optimize both GDSPH and ISPH for this case; here the kernels optimized with GDSPH are given by $O_{N_{smooth}}$ and with ISPH $O_{I,N_{smooth}}$, which was done for $N_{smooth} = 32, 64, 128, 256$. The kernels and the resulting coefficients that were used for the optimization are given in Tables C.2 and C.3.

The kernel function, its derivative and its Fourier transform can be seen in Fig. 3 for the optimized kernels together with C_2 and C_4 kernels. All the resulting kernels have similar amplitude and form as the Wendland $C2$ kernel (except for $O_{I,32}$), but with a minima in its derivative being a bit offset from the $C2$ kernel. Looking at the Fourier transform we can see quite a clear trend with both the $O_{N_{smooth}}$ and $O_{I,N_{smooth}}$ as we increase N_{smooth}. Both O_{32} and $O_{I,32}$ drop the fastest within $k \propto 0 - 3\pi$ but remain higher than the other kernels as we go to larger wave numbers. As we increase N_{smooth} we have a higher power between $k \propto 0 - 3\pi$ but lower for higher wave numbers. We have two resulting kernels with regions of negative Fourier transform. The $O_{I,32}$ kernel, which goes negative in the region $k \propto 4 - 5\pi$, and becomes susceptible to the pairing instability as we go to higher N_{smooth} (we see instability at $N_{smooth} = 128$ and $N_{smooth} = 256$). The O_{256} also has regions of negative Fourier transform at much higher wave numbers ($k \propto 10{-}12\pi$). However, we did not see any sort of instability occurring in this kernel, at least up to $N_{smooth} = 512$. The performances of these kernels are discussed more in Sect. 4.

4. Results

In the following section we discuss the performance of these optimized smoothing kernels on a select number of test cases. First we go through the hydrostatic glass and the Gresho-Chan vortex, both of which we outlined and used for optimization of the kernels in Sects. 3.4.1 and 3.4.2. We then go through four additional tests to see how well the optimized kernels perform in general by testing the Kelvin-Helmholtz instability for two different density contrasts ($\Delta\rho = 2, \Delta\rho = 8$), the classic Sod shocktube test, Evrard collapse and the subsonic blob test. The hydrostatic glass and Gresho-Chan vortex are run for all the optimized kernels. The ones optimized to hydrostatic glass with GDSPH: $O_{glass,32}, O_{glass,64}, O_{glass,128}, O_{glass,256}$. The ones optimized for Gresho-Chan vortex with GDSPH $O_{32}, O_{64}, O_{128}, O_{256}$ and with ISPH $O_{I,32}, O_{I,64}, O_{I,128}, O_{I,256}$. We also compare with a select number of kernel's mentioned in Sect. 2 ($CM_{04}, WU_2, BUH, CM_{05}, C2, C4, C6$). This is done for $N_{smooth} = 32, 64, 128, 256$. For the Kelvin-Helmholtz instability, Sod shocktube, Evrard collapse and subsonic blob test we compare the best-performing kernel at a given N_{smooth} with the popular Wendland C2 and C4 kernel, for both GDSPH and ISPH. The optimized kernels was implemented in both the Gasoline2 code (Wadsley et al. 2017) and the ChaNGa code (Menon et al. 2015). Results are from simulations performed with Gasoline2, but we have also confirmed that ChaNGa gives the same results in many of these cases.

4.1. Hydrostatic glass

The setup of this test case is outlined in Sect. 3.4.1. The system is evolved for a long enough time to reach a relaxed glass state. The quality of the resulting glass is quantified with the SPH errors (E_0 Eq. (3) E_1 Eq. (4)) and the density error (Eq. (46)). These

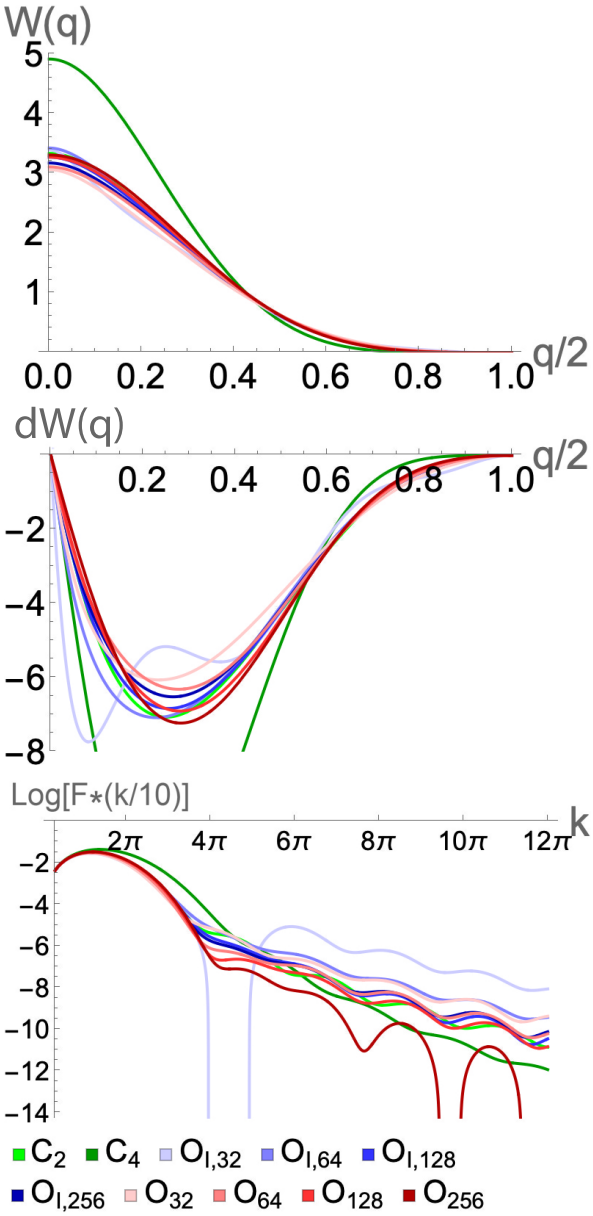

Fig. 3. *Top*: optimized kernel functions for the Gresho-Chan vortex (Sect. 3.4.2), for both GDSPH and ISPH together with the popular Wendland kernels (C_2,C_4). *Middle*: corresponding derivative of these kernel function. *Bottom*: fourier transform of these kernels, scaled by $k/10$. The optimized kernel exhibit a clear trend in it's Fourier transform, where the kernels optimized for higher N_{smooth} emphasizes lower power beyond $k > 3\pi$. The general form of the optimized kernel is quite similar to the Wendland C_2 kernel. In general GDSPH produces lower power than ISPH in the high wave number regime for a given N_{smooth}. The $O_{I,32}$ kernel is the only optimized kernel that exhibits a negative Fourier transform at low wavenumbers. Among the other optimized kernels, only O_{256} has a negative Fourier transform, but this occurs at higher wavenumbers and remains stable to pairing instability up to at least $N_{smooth} < 500$.

are given for all kernels and chosen neighbor numbers in Fig. 4. While the kernel's optimized for the hydrostatic glass perform in average the best considering the cost function (Eq. (45)), we can see that it loses out to some other kernels for specific errors. All the GDSPH optimized kernels for the Gresho-Chan vortex perform fairly well, similar to the BUH, CM_{05} and C_2 kernel for the $\delta\rho_{err}$ and E_0 error. The O kernels does in general have

GDSPH $\delta\rho_{err}$

Kernel	N32	N64	N128	N256
$C2$	0 %	0 %	0 %	0 %
O_{32}	−22.9 %	5.3 %	6.2 %	5.3 %
O_{64}	−3.6 %	7.3 %	13.5 %	10.5 %
O_{128}	12.4 %	−4.5 %	18.5 %	10.5 %
O_{256}	16 %	4.9 %	20.9 %	10.5 %
$O_{I,32}$	211.3 %	143.6 %	364.2 %	455.1 %
$O_{I,64}$	7.7 %	10.2 %	9.8 %	10.5 %
$O_{I,128}$	−4.4 %	5.7 %	8.6 %	13.2 %
$O_{I,256}$	−7.7 %	8.6 %	9.8 %	13.2 %
$O_{glass,32}$	−5.5 %	6.9 %	20.9 %	13.2 %
$O_{glass,64}$	40.5 %	−26.5 %	−6.2 %	−7.9 %
$O_{glass,128}$	38 %	−26.9 %	−6.2 %	−5.3 %
$O_{glass,256}$	22.3 %	−5.7 %	−3.7 %	−7.9 %
CM_{04}	150.4 %	36.7 %	43.1 %	18.4 %
WU_2	54.5 %	−17.1 %	−4.9 %	−10.5 %
BUH	−16 %	2.9 %	1.2 %	0 %
CM_{05}	−16 %	0.8 %	1.2 %	5.3 %
$C4$	219.8 %	47.3 %	30.8 %	52.6 %
$C6$	391.9 %	146.5 %	67.7 %	50 %

E_0

Kernel	N32	N64	N128	N256
$C2$	0 %	0 %	0 %	0 %
O_{32}	−26 %	−3 %	6.8 %	−18.6 %
O_{64}	−8.8 %	−0.8 %	13.7 %	0 %
O_{128}	13.3 %	−6.8 %	29.6 %	0 %
O_{256}	15.5 %	0.8 %	22.8 %	−3.7 %
$O_{I,32}$	193.5 %	128.3 %	357.6 %	320.1 %
$O_{I,64}$	1.1 %	0.8 %	18.2 %	−3.7 %
$O_{I,128}$	−7.2 %	1.5 %	13.7 %	−11.2 %
$O_{I,256}$	−12.7 %	0.8 %	11.4 %	−7.4 %
$O_{glass,32}$	−11.1 %	−1.5 %	27.3 %	−3.7 %
$O_{glass,64}$	42.6 %	−27.3 %	9.1 %	0 %
$O_{glass,128}$	38.7 %	−25.8 %	−2.3 %	−22.3 %
$O_{glass,256}$	21.6 %	−5.3 %	4.6 %	−22.3 %
CM_{04}	130.5 %	13.7 %	11.4 %	−18.6 %
WU_2	49.2 %	−19 %	4.6 %	−22.3 %
BUH	−15.5 %	2.3 %	9.1 %	−11.2 %
CM_{05}	−15.5 %	−3.8 %	4.6 %	−11.2 %
$C4$	210.1 %	53.1 %	52.4 %	22.3 %
$C6$	294.7 %	163.2 %	104.8 %	55.8 %

E_1

Kernel	N32	N64	N128	N256
$C2$	0 %	0 %	0 %	0 %
O_{32}	49 %	−24.2 %	91.6 %	109.8 %
O_{64}	−43.4 %	−59.6 %	49.1 %	27.9 %
O_{128}	−49 %	−67.9 %	13.8 %	3.7 %
O_{256}	−41.1 %	−68.5 %	−20.3 %	−14.9 %
$O_{I,32}$	179 %	−38.6 %	671.4 %	1,721.3 %
$O_{I,64}$	71.9 %	−30.9 %	141.8 %	115.4 %
$O_{I,128}$	−5 %	−43.4 %	83.2 %	26.1 %
$O_{I,256}$	−19.1 %	−51.2 %	59.8 %	37.2 %
$O_{glass,32}$	−42.3 %	−58.6 %	50.3 %	57.7 %
$O_{glass,64}$	−34.3 %	−63 %	−16.2 %	7.4 %
$O_{glass,128}$	−33.6 %	−57.2 %	−29.9 %	−13 %
$O_{glass,256}$	−34.5 %	−76.2 %	−25.1 %	−11.2 %
CM_{04}	100.3 %	3 %	159.8 %	262.4 %
WU_2	−27.3 %	−55.7 %	3 %	0.1 %
BUH	49.2 %	29.9 %	47.9 %	27.9 %
CM_{05}	82.5 %	37.2 %	146 %	111.6 %
$C4$	367 %	0.1 %	35.9 %	7.4 %
$C6$	1,247.5 %	152.4 %	105.9 %	20.5 %

Fig. 4. Relative increase or decrease of the $\delta\rho_{err}$ (*top*), E_0 (*middle*), and E_1 (*bottom*) of the hydrostatic glass when relaxed for different kernels in respect to the corresponding error of the C_2 kernel. The simulations here were performed with only the GDSPH method. The terms N32, N64, N128, and N256 refer to the number of neighbors: N_{smooth} = 32, 64, 128, 256, respectively. The $\delta\rho_{err}$, E_0 and E_1 of the C2 kernel for each N_{smooth} is $\delta\rho_{err} = 5.539 \times 10^{-3}, 3.747 \times 10^{-3}, 1.243 \times 10^{-3}, 5.860 \times 10^{-4}$, $E_0 = 2.763 \times 10^{-3}, 2.024 \times 10^{-3}, 6.797 \times 10^{-4}, 4.253 \times 10^{-4}$, and $E_1 = 2.210 \times 10^{-2}, 1.484 \times 10^{-2}, 2.557 \times 10^{-3}, 8.305 \times 10^{-4}$. The optimized kernels can in general be seen to drastically reduce the E_1 error relative to the C2 kernel. For the glass optimized kernel's we can also see a general decrease in both $\delta\rho$ and E_0 for their given N_{smooth}.

GDSPH $L_{1,error}$

Kernel	N32	N64	N128	N256
$C2$	0 %	0 %	0 %	0 %
O_{32}	−18.4 %	−2.2 %	−2.9 %	6.8 %
O_{64}	−6.5 %	−41 %	−19.2 %	−1.6 %
O_{128}	5.1 %	−36.2 %	−31.1 %	−8 %
O_{256}	10.9 %	−37.5 %	−28.7 %	−17.7 %
$O_{I,32}$	91.5 %	182.6 %	352.3 %	420.6 %
$O_{I,64}$	2.6 %	−18.7 %	−0.3 %	13.9 %
$O_{I,128}$	−2.6 %	−34.9 %	−9.5 %	3.6 %
$O_{I,256}$	−5.7 %	−38.2 %	−15.4 %	−0.7 %
$O_{glass,32}$	−10.3 %	−37.6 %	−15.5 %	1.8 %
$O_{glass,64}$	17.4 %	2.3 %	3.4 %	−0.6 %
$O_{glass,128}$	17.7 %	−2.7 %	−6 %	−5.1 %
$O_{glass,256}$	8.9 %	−5.6 %	−2.9 %	−3.6 %
CM_{04}	87.2 %	94.5 %	142.6 %	123 %
WU_2	41 %	22.9 %	28.9 %	14.4 %
BUH	−1.4 %	5.8 %	7.9 %	8.8 %
CM_{05}	−10.7 %	22.7 %	15.2 %	20.8 %
$C4$	79.7 %	4.2 %	7.7 %	1.4 %
$C6$	92.2 %	101.3 %	41.7 %	12.3 %

ISPH $L_{1,error}$

Kernel	N32	N64	N128	N256
$C2$	0 %	0 %	0 %	0 %
O_{32}	−10.5 %	−18.6 %	−2.7 %	−1.1 %
O_{64}	−8.3 %	−8.4 %	−7.1 %	−6.9 %
O_{128}	−3.7 %	1.6 %	8 %	−1.4 %
O_{256}	−1.2 %	1.5 %	1.7 %	−1.3 %
$O_{I,32}$	−30.1 %	14.5 %	107.6 %	173.7 %
$O_{I,64}$	−12.1 %	−21.4 %	−6.7 %	−3.6 %
$O_{I,128}$	−8 %	−15.5 %	−11.5 %	−7.3 %
$O_{I,256}$	−9.3 %	−15.9 %	−11.5 %	−8.3 %
$O_{glass,32}$	−10.5 %	−12.3 %	−6.7 %	−5.4 %
$O_{glass,64}$	4.8 %	23.6 %	96.3 %	130.4 %
$O_{glass,128}$	0.7 %	18 %	85.1 %	109 %
$O_{glass,256}$	1.9 %	9.5 %	32.9 %	38.9 %
CM_{04}	19.2 %	92.4 %	240.4 %	315.8 %
WU_2	1.3 %	44.5 %	137.4 %	191.7 %
BUH	−0.1 %	−4.6 %	−1.8 %	−1.3 %
CM_{05}	−5.1 %	−12.4 %	2.1 %	1 %
$C4$	21.3 %	41.3 %	2 %	−5.6 %
$C6$	51.7 %	82.7 %	39.5 %	1.2 %

Fig. 5. Relative increase or decrease of the $L_{1,error}$ of the Gresho-chan vortex at $t = 1$ and $n_x = 64$ for different kernels in respect to the $L_{1,error}$ of the C_2 kernel. The terms N32, N64, N128, and N256 refer to the number of neighbors: $N_{smooth} = 32, 64, 128, 256$. *Left*: simulations done with the GDSPH method. *Right*: simulations done with the ISPH method. The $L_{1,error}$ of the C2 kernel for each N_{smooth} is $L_{1,error} = 1.49375 \times 10^{-1}, 9.57342 \times 10^{-2}, 4.70888 \times 10^{-2}, 2.99816 \times 10^{-2}$ for GDSPH and $L_{1,error} = 9.34635 \times 10^{-2}, 4.66192 \times 10^{-2}, 2.64744 \times 10^{-2}, 2.22268 \times 10^{-2}$ for ISPH. An improvement for the optimized kernel at a given N_{smooth} across all N_{smooth} can be seen for both methods, where the biggest improvements are for the $N64$ and $N128$ for GDSPH and $N32$ and $N64$ for ISPH. Of the non-optimized kernels, we observed that the C_2 kernel performs best, except for $N_{smooth} = 32$ (GDSPH), $N_{smooth} = 32, 64$ (ISPH), where CM_{05} performs better and C_4 performs slightly better at $N_{smooth} = 256$ (ISPH).

smaller E_1 errors than the standard kernels. The only exception is the O_{32} kernel that has higher E_1 error but has the smallest density and E_0 error out of all the kernels for N_{32}. The WU_2 kernel performs very well for higher neighbor numbers and is close to the results of the optimized kernel $O_{glass,128}$ and $O_{glass,256}$. The biggest improvement that can be seen when comparing the O_{glass} and O kernels with the Wendland $C2$ kernels, is in the E_1 errors that are significantly reduced. We can see that the $O_{I,32}$ kernel performs very badly when relaxing to a hydrostatic glass with the GDSPH gradient operator, even at $N_{smooth} = 32$. This is, however, not the case when relaxing with ISPH at $N_{smooth} = 32$, indicating that glass relaxation is different when using ISPH.

4.2. Gresho-Chan vortex

The setup of this test case is outlined in Sect. 3.4.1. We run the Gresho-Chan vortex up to $t = 3$. We also perform a convergence study, to see if the optimized kernels still perform well as we increase/decrease the resolution, as the optimization was performed at $n_x = 64$ resolution. For this convergence study we only compare with the Wendland C_2 and C_4 kernels. The optimization of the $O_{N\,smooth}$ kernels was done at $t = 1$, we can see the result of the $L_{1,error}$ (see Eq. (50)) at $t = 1$ in Fig. 5 for both GDSPH and ISPH. From this table we can see that the optimized kernels for the Gresho-Chan vortex perform the best for their given N_{smooth}. The difference in $L_{1,error}$ is more pronounced for lower N_{smooth} and for the GDSPH method compared to the ISPH method. A comparison of the azimuthal velocity structure ($n_x = 64, t = 3$) between the optimized kernel and the Wendland C_2 and C_4 kernel can be be seen in Fig. 6 for GDSPH and ISPH. For GDSPH we can see that the optimized kernel mitigates the decay of the vortex structure (in particular for $N_{smooth} = 64$). For ISPH the effect of the optimized kernels can mainly be seen as less noise in the vortex structure, while the general decay is similar across all the kernels (as this is highly dependent on the linear gradient estimate). A convergence study at $t = 3$ of the $L_{1,error}$ can be seen in Fig. 7 for GDSPH and Fig. 8 for ISPH. Looking at the convergence study, we can see that the improvement of the optimized kernels in general stays true for higher and lower resolution.

Looking at the GDSPH results, we can see that while the O_{128} kernel proved to be the best at $t = 1$, this is no longer true at $t = 3$ and $N_{smooth} = 128$, where the O_{256} performs better across all resolutions at this time. And looking at the result at the earlier time of $t = 1$, the O_{128} just barely performed better at $N_{smooth} = 128$ $n_x = 64$, but worse than O_{256} for higher resolution. We can also see that the O_{256} performs better than the O_{64} at $N_{smooth} = 64$ for the higher resolutions($n_x = 128, 256$) at both $t = 1$ and $t = 3$. The improvement seen in $N_{smooth} = 64$ for GDSPH is the largest improvement that we see in across all our Gresho-Chan cases, where the optimized kernels effectively half the $L_{1,error}$ compared to the Wendland C_2 and C_4 kernels, resulting in errors similar to $N_{smooth} = 128$ errors of the Wendland kernels (at both $t = 1$ and $t = 3$). This is quite clearly seen in Fig. 6, when comparing the O_{64} $N_{smooth} = 64$ vortex structure with the $N_{smooth} = 128$, C_2 and C_4 vortex structure. A quite surprising result is that the C_4 kernel performs better than the C_2 kernel for $N_{smooth} = 64$, while performing worse for $N_{smooth} = 32, 128$, where at $N_{smooth} = 32$ it performs drastically worse. For $N_{smooth} = 32$, we can see that the C_2 kernel performs relatively well, but with O_{32} being about a 20−30% improvement at $t = 1$ and the only kernel with some sort of vortex structure at $t = 3$ (as can be seen in Fig. 6).

Looking at the result from ISPH, we can see a much less difference between all the different kernels, especially at high neighbor number ($N_{smooth} = 128, 256$). At $t = 3$ and $N_{smooth} = 256$ we can see that the C_4 kernel actually performs better than all other kernels, including the optimized kernel $O_{I,256}$. This is quite surprising, as the $O_{I,256}$ performs better at earlier times, and looking at the E_0 errors at $t = 3$ in Fig. 9, we can see that they are higher for C_4 compared to all other kernels. This likely means that C_4 has smaller higher order errors, compared to the other kernels. The $O_{I,32}$ kernel performs significantly better than all

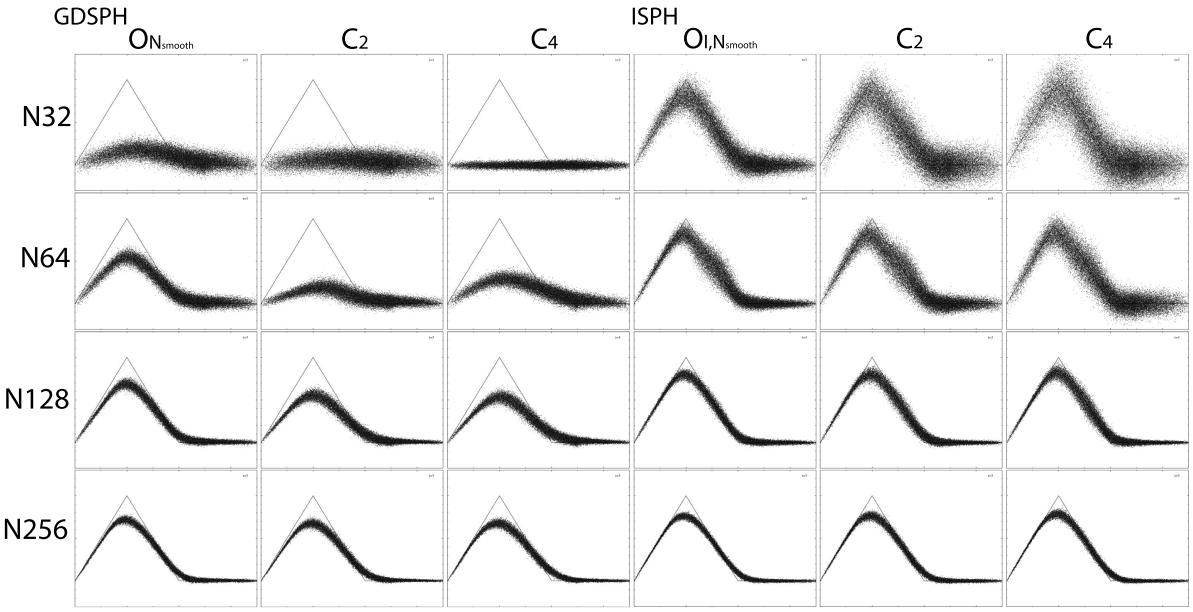

Fig. 6. Azimuthal velocity in the radial direction of the Gresho-Chan vortex at $t = 3$. Here the resolution is $n_x = 64$ and simulation was performed with GDSPH (*left three columns*) and ISPH (*right three columns*). We compare the optimized kernel at a given N_{smooth} to the popular Wendland C_2 and C_4 kernels, and vary the number of neighbors in each row ($N_{smooth} = 32, 64, 128, 256$), referred here to N32, N64, N128, N256. The solid black line show the analytical solution and the black dots show the results from the simulation. The X axis (cylindrical radius) extends from 0.0 to 0.7, and the vertical extent from the top to the bottom of the analytical solution is 1. We can see a significant improvement in the optimized kernels for the GDSPH method for $N_{smooth} = 32, 64, 128$, where the vortex structure becomes more pronounced. Overall less noise can be seen for all N_{smooth}. The ISPH method captures the overall vortex structure well for all N_{smooth}, due to exact linear gradients. But we can see significant improvement in the noise level of the optimized kernels, compared to the C_2 and C_4 kernels.

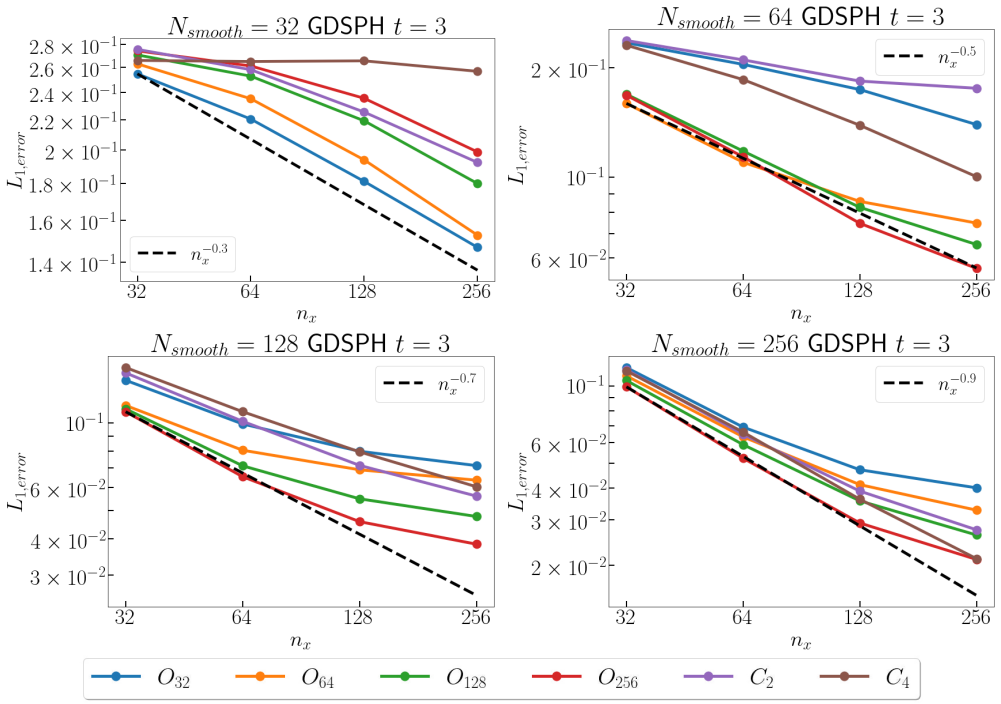

Fig. 7. Convergence study of the Gresho-Chan vortex with the GDSPH method for $N_{smooth} = 32, 64, 128, 256$. Here we plot the $L_{1,error}$ over resolution elements in the x direction (n_x) at $t = 3$. Significant improvements can be seen for all optimized kernels compared to the Wendland C2 and C4 kernels. The kernels were optimized in respect to an earlier time $t = 1$ and we can see that O_{256} have an improved result over O_{128} for $N_{smooth} = 128$ at this later time, even beating out O_{64} for $N_{smooth} = 64$ at higher resolution ($n_x > 128$). The $C4$ kernel exhibits slightly better convergence slope at high N_{smooth} and $n_x > 128$, likely due to it's higher order properties (fewer $O(h^2)$ errors).

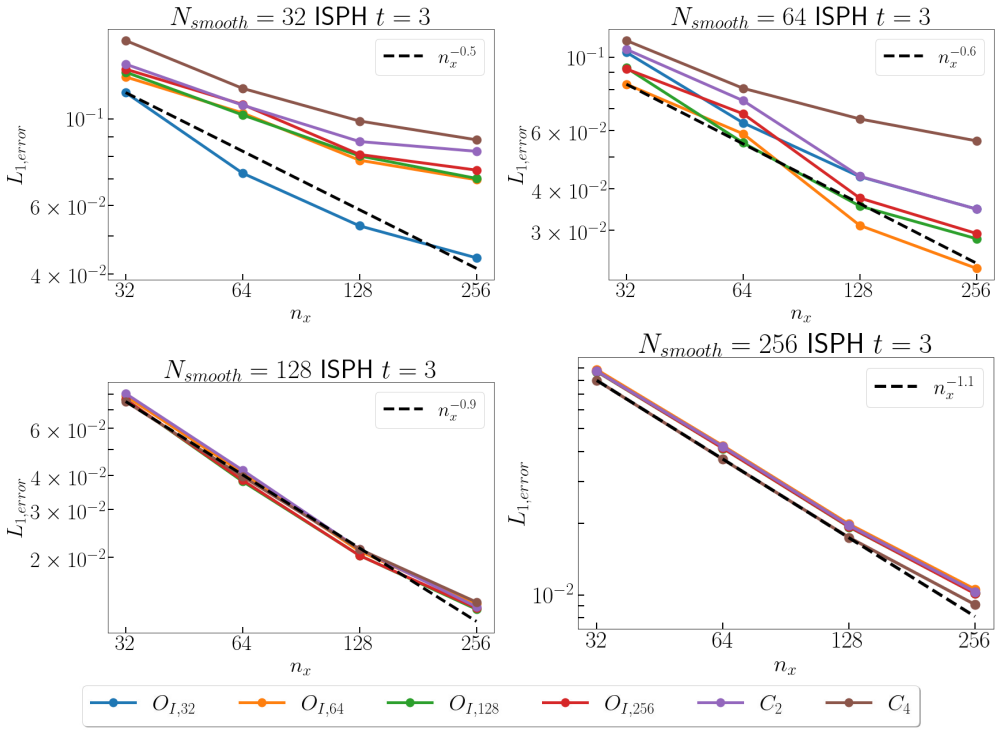

Fig. 8. Convergence study of the Gresho-Chan vortex with the ISPH method for $N_{smooth} = 32, 64, 128, 256$. Here we plot the $L_{1,error}$ over the resolution elements in the x direction (n_x) at $t = 3$. Significant improvements can be seen for the optimized kernels at $N_{smooth} = 32, 64$ compared to the Wendland C2 and C4 kernels. While at higher N_{smooth}, there is much less difference between all the kernels. The kernels were optimized in respect to an earlier time, $t = 1$, and we observed that C_4 performs better at this later time for $N_{smooth} = 256$ than all the optimized kernels at all resolutions. This is surprising, as E_0 errors of the optimized kernels are much lower than the C_4 kernel (see Fig. 9), which might indicate that long-term behavior at this N_{smooth} is heavily determined by the higher order errors ($O(h^2)$).

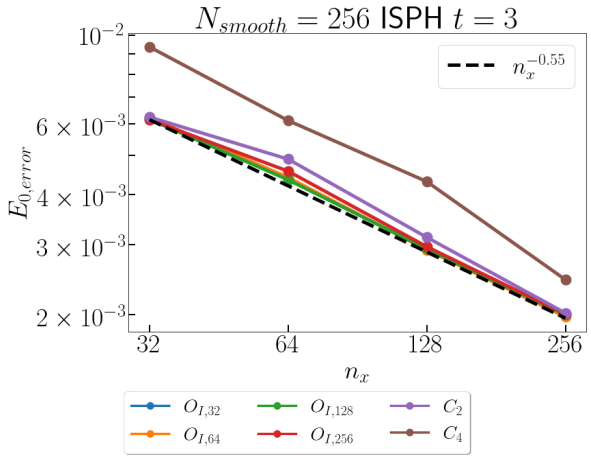

Fig. 9. Convergence study of the Gresho-Chan vortex with the ISPH method for $N_{smooth} = 256$. Here we plot the E_0 error versus the number of resolution elements in the x direction (n_x) at $t = 3$. Much lower zeroth order errors for the optimized kernel and C_2 kernel can be seen compared to the higher order C_4 kernel. Note here that even though being zeroth order (dependent on local particle order), the error still decreases with increasing resolution ($n_x^{-0.55}$) because the particle distribution becomes more uniform at higher resolution.

the other kernels at $N_{smooth} = 32$ and is the only optimized kernel with negative Fourier transform at relatively low wave number, which means that it becomes unstable to pairing instability at higher neighbor numbers (above $N_{smooth} = 64$). The Wendland C_4 kernel is by far the worst kernel at lower neighbor numbers ($N_{smooth} = 32, 64$) for ISPH. For $N_{smooth} = 64$ at both $t = 1$ and $t = 3$, the $O_{I,64}$ produce a significant improvement compared to the Wendland kernels.

The convergence scaling seems to be similar between kernels, but with some experiencing a flattening at higher resolution. The C_4 kernel seems to scale a bit better than other kernels at higher resolution, though still being worse in general. The scaling will also depend on the given dissipation scheme/parameters used. We have also run the Gresho-Chan vortex with different dissipation parameters ($\alpha = 0.01, 0.05, 0.1$) and the relative results seen in this section remains the same.

4.3. Kelvin Helmholtz

The Kelvin-Helmholtz (KH) instability occurs when there is a shearing flow between two fluid interfaces. The instability quickly generates a series of rolling waves at the interface between the two fluids, which eventually breaks down into turbulence, mixing the two fluids. The instability plays a significant role in all kinds of fluids and is a crucial component for any hydrodynamical code to model, as it effectively allows two fluids to mix and generate turbulence. However, traditional SPH methods have been known to struggle with triggering the KH instability, particularly when there are significant density contrasts between the fluids. In this paper we model the KH instability using a similar setup to the ones outlined in McNally et al. (2012); Lecoanet et al. (2016); Tricco (2019), using a thin periodic box ($L = [1, 1, 2\frac{\sqrt{6}}{n_x}]$), here n_x is the average number of particles in the x direction, and the z boundary is set to be roughly 24 particle spacings. A smoothing function was applied

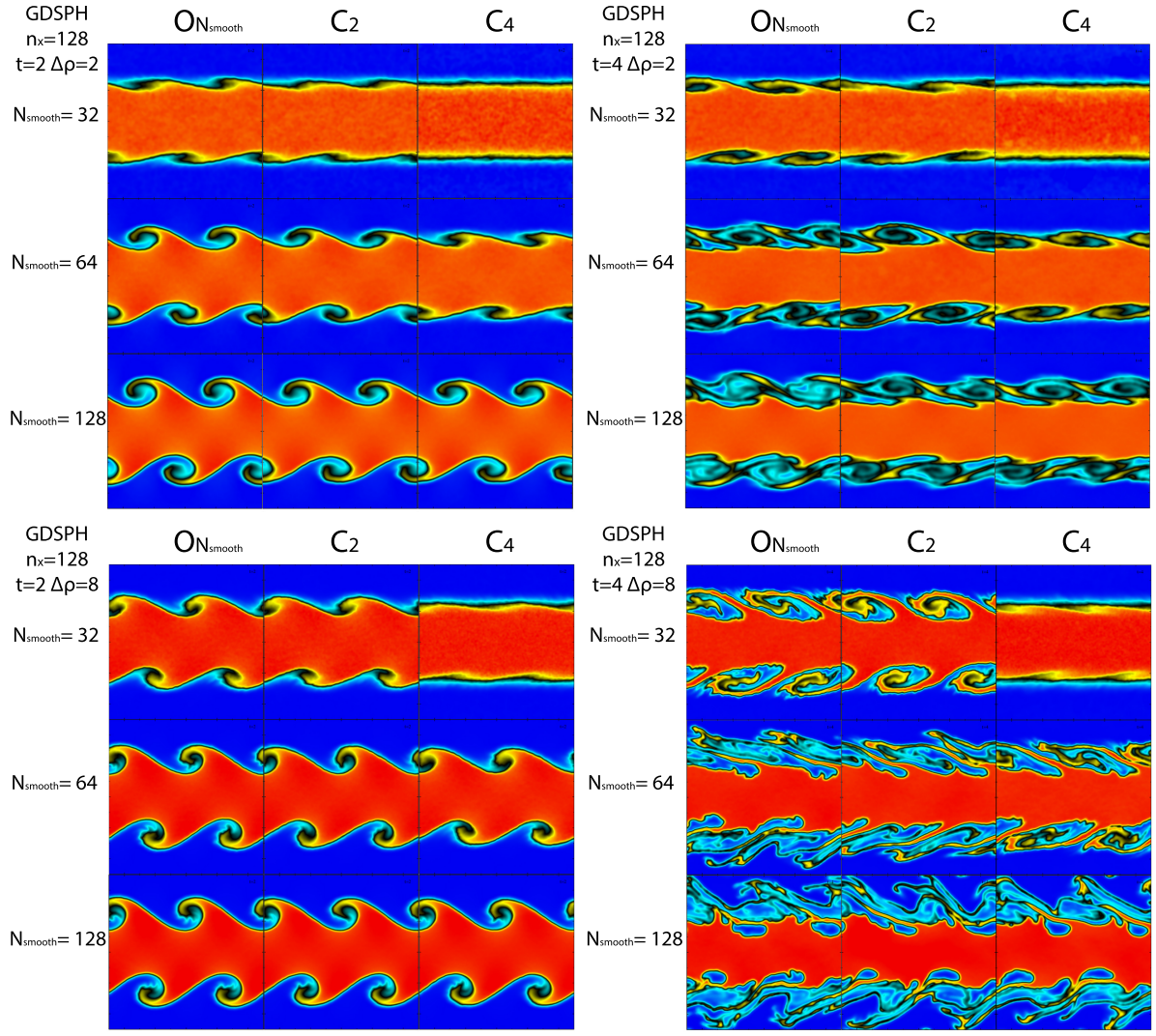

Fig. 10. Density rendering of the Kelvin-Helmholtz instability at t=2 (*left column*) and t=4 (*right column*) for an initial density contrast of $\Delta\rho = 2$ (*top panels*) and $\Delta\rho = 8$ (*bottom panels*) with the GDSPH method. The X and Y axes have a length of one. Here one can see a comparison in the development of the instability between the optimized kernel at a given N_{smooth} and the popular Wendland C_2 and C_4 kernels. We observed a clear improvement in the optimized kernel, reproducing a similar behavior of the C_4 kernel at double the neighbor count (N_{smooth}). For the high density contrast $\Delta\rho = 8$, there are only minor differences between the different kernels at $N_{smooth} = 128$. These can be compared to the higher resolution cases given in Appendix A.

to the density and the shearing velocity accordingly:

$$f_R = 0.5 * \left(\tanh \frac{y + y_{inner}}{\delta} - \tanh \frac{y - y_{inner}}{\delta}\right), \tag{51}$$

$$\rho = \rho_{outer} + f_R(\rho_{inner} - \rho_{outer}), \tag{52}$$

$$v_x = v_{x,outer} + f_R(v_{x,inner} - v_{x,outer}), \tag{53}$$

$$v_y = 0.01 \sin(4\pi x). \tag{54}$$

Here we set $\rho_{outer} = 1$, $\delta = 0.025$, $v_{x,outer} = -0.5$, and $v_{x,inner} = 0.5$. We simulated the KH instability with two different density contrasts ($\rho_{inner} = \Delta\rho = 2$ and $\rho_{inner} = \Delta\rho = 8$). The pressure was initially uniform ($P_0 = \frac{1}{\Gamma\mathcal{M}^2}$), where Γ is the adiabatic index and $\mathcal{M}$ is the Mach number. We used an adiabatic equation of state with adiabatic index of $\Gamma = 5/3$ and a Mach number of $\mathcal{M} = \sqrt{6}/5$. The IC is setup using the IC glass generator as described in Sect. 3.2 for all the different kernels. We ran the simulation until $t = 4$ for each of the optimized kernels at their specific $N_{smooth} = 32, 64, 128, 256$ and SPH method. We then compared these kernels with the Wendland C_2 and C_4 kernel.

In Fig. 10 we show the rendered structure of the KH instability for the GDSPH $\Delta\rho = 2$ and $\Delta\rho = 8$ at $t = 2$ and $t = 4$. Due to the $N_{smooth} = 256$ simulations producing near identical results to the $N_{smooth} = 128$ we only show the results of $N_{smooth} = 128$. For both the low and high density contrast cases, we can see a clear improvement in the growth of the instability for the optimized kernels, producing similar structure as the C_4 kernel at half the N_{smooth}. In Fig. 12 we show the amplitude growth of the $k = 2\pi$ mode, using the same procedure as outlined in McNally et al. (2012), which more quantitatively show the improvement of the optimized kernel for GDSPH.

In general, the difference between the kernels in the ISPH method is very minor for the KH case, across the different N_{smooth}, density contrasts and times. The only exception is for $N_{smooth} = 32$ at $t = 4$ (see Fig. 11), where in the $\Delta\rho = 2$ case the C_4 experiences significant non-conservation of angular

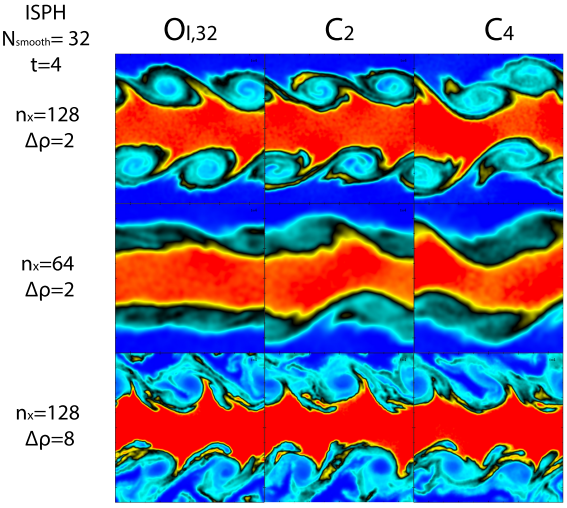

Fig. 11. Density rendering of the Kelvin-Helmholtz instability at $t = 4$ for $N_{smooth} = 32$ with the ISPH method. The X and Y axes have a length of one. Here one can see a comparison in the development of the instability between the optimized kernel $(OI, 32)$ and the popular Wendland C_2 and C_4 kernels. We have selected three cases where discernible differences can be seen ($n_x = 128, \Delta\rho = 2$; $n_x = 64, \Delta\rho = 2$; $n_x = 128, \Delta\rho = 8$). For the $\Delta\rho = 2$ cases we observed that the C_2 and C_4 kernels exhibit more angular momentum errors (inherent to ISPH) compared to the $O_{I,32}$ kernel, where the inner stream becomes significantly twisted, deviating from the high resolution and high N_{smooth} results (see Appendix A). For the high density case $n_x = 128, \Delta\rho = 8$, we observed that C_4 is unable to generate the same vortex structure in the upper plane of the stream and exhibit some asymmetry around the plane.

momentum, leading to the flow rotating (still capturing the rolling vortex waves). For $\Delta\rho = 8$, $t = 4$ the optimized kernel, produces a vortex structure (seen at higher N_{smooth} for all kernels), while the C_4 kernel fails to capture the vortex. The optimized kernel also produces sharper density structure compared to C_2 kernel.

We have also performed more comparisons and a resolution study between GDSPH and ISPH for the KH test, which can be found in Appendix A.

4.4. Sod shock tube

Here we test the classic Sod shock tube (Sod 1978), where a fluid medium is initially divided into two separate regions with different pressures and densities. The initial conditions are $\rho_{left} = 1$, $P_{left} = 1$ for $x < 0$ and $\rho_{right} = 0.25$, $P_{right} = 0.1795$ for $x > 0$. We use an adiabatic equation of state with adiabatic index of $\Gamma = 5/3$. At the start of the simulation, we imagine removing a wall between the two regions creating a discontinuous state between the fluid's generating a shockwave. The produced shockwave has three distinct regions, the right-moving shock front, the left-moving rarefaction wave, and the contact discontinuity. Here there are once again many ways to setup the initial distribution of particles. This case is particularly difficult due to the discontinuous density profile, as it is not possible to follow a discontinuous solution with the smooth density estimate. That is, even if two different density glasses or lattices are set up, the density should still be smoothed at the interface. One could either try one's best to relax a glass following the discontinuous solution or smooth the initial shock by a tiny amount (proportional to the smoothing length). An argument against the complete discontinuous solution (in density) would be that it would rarely happen in a dynamic SPH simulation, as it would always be accompanied with some smoothing. Other quantities, such as velocities and thermal energy, of course occur discontinuously in simulations. We chose to smooth the discontinuity on the scale of the smoothing length. We used a thin periodic 3D box to approximate the 1D test, with $L_x = 2.0$ and $L_y = L_z$ set to be roughly 16 particle spacings in the low density region. We used $L_x = 2.0$ due to using periodic boundary conditions such that the shock generated at the periodic boundary does not interact with the central shock. We used a resolution of around $n_x = 64 \times 2$ in the initial high density region (the factor of two is to account for the box extension). We used a constant $\alpha = 1$ and $\beta = 2$ for the artificial viscosity.

As the velocity shows the biggest difference, we highlight the velocity structure between the different kernels in Fig. 13 for $N_{smooth} = 64$. From this figure we can see that the optimized kernel's exhibit less noise in the post-shock region compared to the Wendland C2 and C4 kernels for both GDSPH and ISPH. Other than that, there is not much difference between the kernels in this case.

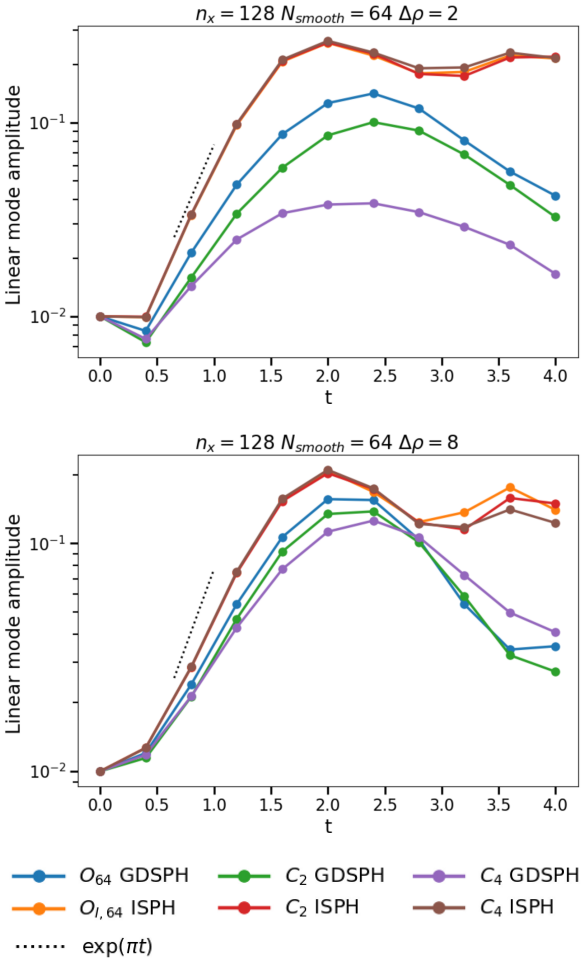

Fig. 12. Amplitude of the $k = 2\pi$ mode over time for different smoothing kernels and density contrasts: $\Delta\rho = 2$ (*top*) and $\Delta\rho = 8$ (*bottom*). The ISPH method shows little difference between the different smoothing kernels, capturing a growth rate of roughly $\gamma \propto \pi$ in $\Delta\rho = 2$ case and a slightly lower growth rate for $\Delta\rho = 8$ (as expected). The GDSPH method of $N_{smooth} = 64$ struggles to capture the same growth rate as the ISPH method, but we observed that the optimized kernel improves the growth rate of the KH mode for both $\Delta\rho = 2$ and $\Delta\rho = 8$.

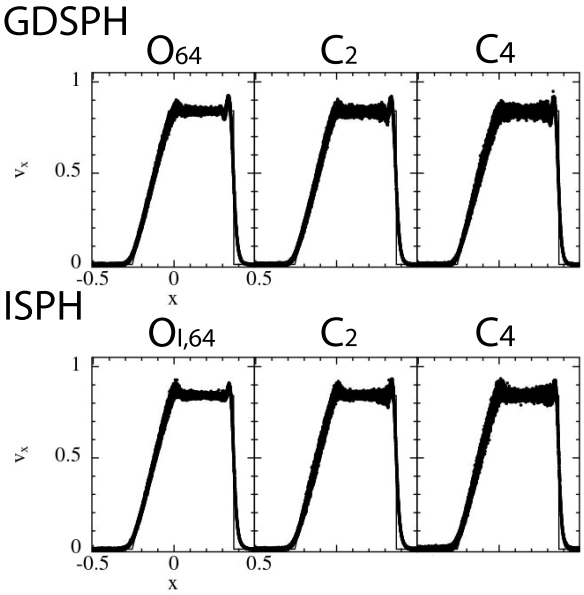

Fig. 13. Velocity structure of the Sod-shocktube test at $t = 0.2$ performed at $n_x = 64$ and $N_{smooth} = 64$. The thin black line shows the exact solution. One can see that the optimized kernel exhibit less noise in the post-shock region compared to the C2 and C4 kernel.

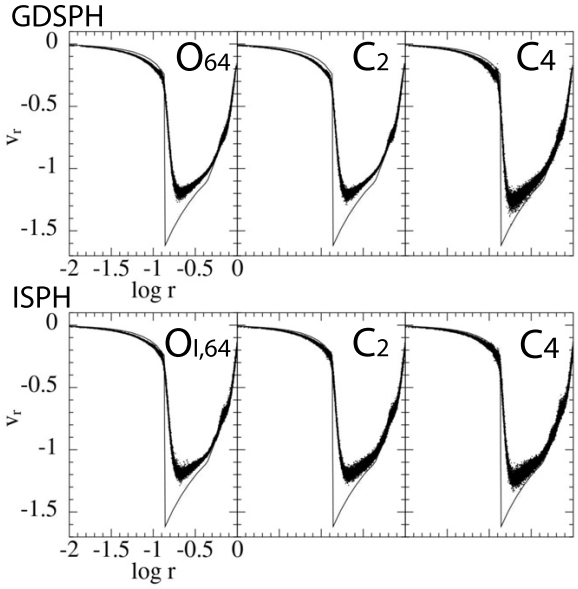

Fig. 14. Velocity structure of the Evrard collapse at $t = 0.8$ performed with 28000 particles and $N_{smooth} = 64$. The thin black line shows the reference solution from a 1D PPM simulation with 8192 grid points (Mandal et al. 2023). One can see that the optimized kernel and the C2 kernel exhibit less noise in the post-shock region compared to the C4 kernel.

4.5. Evrard collapse

The Evrard collapse test (Evrard 1988; Steinmetz & Mueller 1993; Wadsley et al. 2017; Mandal et al. 2023), simulates the collapse of adiabatic gas, which include a high Mach shock and a cold pre-shock infall. The density is defined as

$$\rho(r) = \frac{M(R)}{2\pi R^2 r}, \tag{55}$$

where the initial cloud radius is $R = 1$ and the initial cloud mass is $M = 1$. The internal energy is $u = 0.05GM/R$ and we use an adiabatic index of $\gamma = 5/3$. We setup the initial density using our IC generator, using roughly 28 000 particles. As this is an open boundary IC, we need to assume an initial low density medium for the IC generator to fit against. This is because the IC generator uses only the SPH forces to fit the density profile, this requires a periodic boundary and density to be defined everywhere. After the IC generator has fitted the density profile, we remove all particles beyond $R = 1$ and ensure that particle mass sum up to $M = 1$.

The biggest difference in the Evrard collapse (at $t = 0.08$) between the two methods GDSPH and ISPH and the different kernels can be seen in the noise level, similar to the Sod shocktube test. In Fig. 14 we can see the velocity structure of the two SPH methods and kernels. The noise level of C_4 is slightly larger than the C_2 and the optimized kernels. With C_4 having a slightly sharper shock front than the other two.

4.6. Subsonic blob test

The blob test consist out of a high density spherical cloud at rest within a uniformly flowing low density medium. As the high density blob is accelerated by the surrounding medium, fluid instabilities such as Kelvin-Helmholtz and Rayleigh-Taylor instabilities disrupts the cloud and mixes it with the surrounding medium. The original blob test of Agertz et al. (2007), models a wind with a super sonic Mach number of $\mathcal{M} = 2.7$. However, we have seen from the Sod shock tube and Evrard collapse tests, that the quality of the smoothing kernel does not have much effect on high Mach tests. Particle noise and kernel errors are more relevant for subsonic flows, and as such we decided to instead model a subsonic blob test with a Mach number of $\mathcal{M} = 0.5$ to test the effect of the kernels on the mixing in the subsonic regime. The domain is periodic, with dimensions (10,10,40) in units of cloud radius. The cloud was put in pressure equilibrium and has a density ratio of $\Delta\rho = \rho_{cloud}/\rho_{wind} = 100$. The wind velocity was set by the Mach number $v_{wind} = \mathcal{M}c_s$, and we defined the characteristic cloud crushing time to be $t_{crush} = \sqrt{\Delta\rho}R_{cloud}/v_{wind}$ and ran the simulation for $10t_{crush}$. In our analysis of the blob test, we defined the cloud mass to be $M_{cloud} = M(\rho > \rho_{cloud}/3)$ and the intermediate temperature mass as in Braspenning et al. (2023); Sandnes et al. (2025):

$$M_{mix} = \sum m_i \Big(\log(T_{mix}) - 0.25\log(\Delta\rho) < \log T_i < \log(T_{mix}) + 0.25\log(\Delta\rho) \Big). \tag{56}$$

Here, m_i is mass of the particle, T_i is the temperature of the particle, and T_{mix} is the geometric mean temperature: $T_{mix} = \sqrt{T_{cloud}T_{wind}} = T_{cloud}\sqrt{\Delta\rho}$.

From the top panel of Fig. 16, we can see that all models disrupts the cloud and reduces the high density mass in a few cloud crushing times (3–5 t_{crush}). However, it is clear from the rendering of the simulation at $t = 5t_{crush}$ in Fig. 15 that the mixing of the gas with the surrounding medium differs between the different SPH methods and smoothing kernels. This can also be seen in the intermediate temperature mass evolution, where initially this mass increase is similar in all models, but the subsequent mixing of this mass with the warm medium evolves very differently. ISPH mixes the cloud more aggressively than GDSPH, where for ISPH at $t = 5t_{crush}$ most of the intermediate temperature mass have mixed with the surrounding medium, while

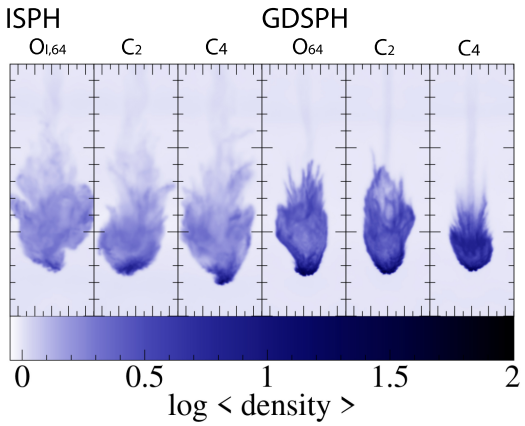

Fig. 15. Density rendering of the subsonic blob test ($\Delta\rho = 100$, $\mathcal{M} = 0.5$ $R_{cloud} = 0.1$) at $t = 5t_{crush}$ with a resolution of $n_x = 64$ in the low density medium. The three left columns depict the cloud state with the ISPH method and the kernels $(O_{I,64}, C_2, C_4)$, and three right columns show the GDSPH method and the kernels (O_{64}, C_2, C_4). The X axis has a length of one, and Y axis has a length of three. One can see that the ISPH method disrupts and mixes the cloud more aggressively than the GDSPH method. We also observed that we increase mixing with the optimized kernels compared to the C_2 and C_4 kernel in both the GDSPH and ISPH method.

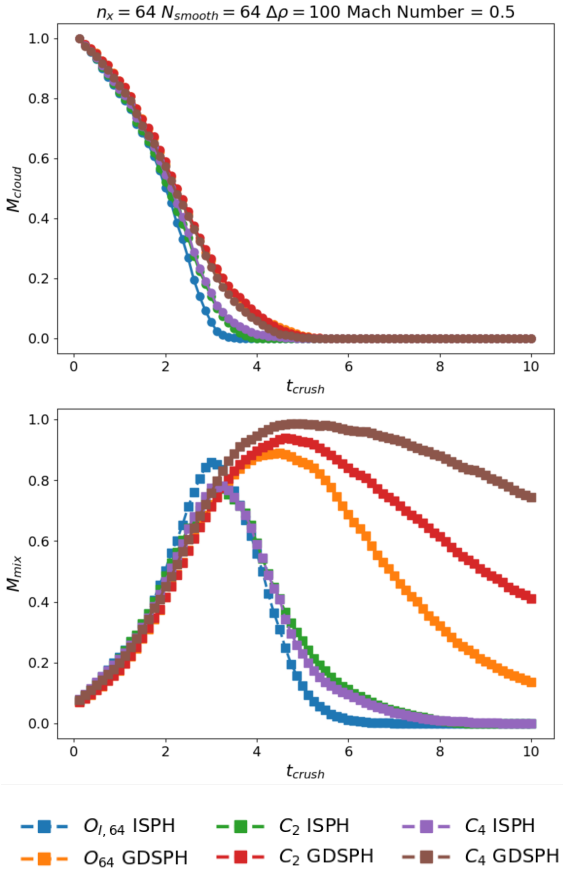

Fig. 16. Time evolution of the cloud mass ($M(\rho > \frac{1}{3}\rho_{cloud})$; *top column*) and the mass of intermediate temperature gas (Eq. (56); *bottom column*). Once can see that it takes about the same time for the cloud mass to be lost in all the different modules, though with a slight faster decrease in the ISPH modules, with the fastest decrease being with the $O_{I,64}$ kernel + ISPH. However, the amount of mixing in intermediate temperature gas differs quite a bit, where the optimized kernels increases the amount of mixing in both the ISPH and GDSPH models. The mixing results are reminiscent of the KH mode growth results shown in Fig. 12.

GDSPH have majority of the initial cloud mass in this intermediate temperature. The result of the intermediate temperature mass, is reminiscent of the KH mode amplitude growth dependency that we saw in Fig. 12.

5. Discussion

In this paper we have generated a set of new optimized kernels through the linear combination of other positive-definite kernels whose coefficients were optimized for the GDSPH method for the hydrostatic glass, the Gresho-Chan vortex, and for the ISPH method for the Gresho-Chan vortex. We find that the kernels optimized for the hydrostatic glass perform significantly worse for dynamical cases. We find that all kernels that had regular large oscillations in their Fourier transform performed worse for the Gresho-Chan vortex. The kernels produced by optimizing to the Gresho-Chan vortex all produce a quite steady decreasing Fourier transform with very slight oscillatory behavior and with a very similar functional form to that of the C_2 kernel. However, compared to the C_2 kernel, the majority of optimized kernels have a Fourier transform that decreases more rapidly with wave numbers (past $k > 3\pi$). We observed that in general, the Fourier transform decreases faster past $k > 3\pi$ for the kernels optimized at a higher neighbor number, and it decreases faster before $k < 3\pi$ for the kernels optimized at a lower neighbor number. Having a lower power at a high wave number seems to be important for higher neighbor numbers, while a lower power at low wave numbers is important for smaller neighbor numbers.

The new optimized kernels produce significant improvements for the GDSPH method, showing an improvement for the Gresho-Chan vortex ($n_x = 64$ $t = 1$) $L_{1,error}$ of $-20, -41, -31$, and -17% relative to the $L_{1,error}$ of the C_2 kernel for $N_{smooth} = 32, 64, 128, 256$, respectively. The optimized kernel for $N_{smooth} = 64$ gives roughly the same $L_{1,error}$ as that of $N_{smooth} = 128$ for the C_2 kernel. At the later time, we observed that the O_{256} kernel performs better than O_{128} for all n_x and even better than O_{64} at higher resolution ($n_x = 128, 256$), as O_{256} performs quite well even at $t = 1$ for both $N_{smooth} = 64$ and $N_{smooth} = 128$. One could use O_{256} for all $N_{smooth} > 64$. The C_4 kernel has the steepest improvement for $N_{smooth} = 256$, remaining near linear with resolution, while O_{256} starts to flatten out beyond $n_x = 256$. This is likely due to the fact that O_{256} reduces zero- and first-order errors a lot, while C_4 reduces higher-order errors more than O_{256}. For ISPH, we also observed improvements of the $L_{1,error}$ in the Greso-Chan vortex, which are $-30, -21, -11$, and -8% relative to the $L_{1,error}$ of the C_2 kernel for $N_{smooth} = 32, 64, 128, 256$, respectively. This is visible mainly in a reduction of noise. At the later time, there is much less difference between the kernels at the higher $N_{smooth} = 128, 256$, while a significant improvement can still be seen for $N_{smooth} = 32, 64$. The C_2 kernel is in general the best-performing smoothing kernel from the non-optimized kernels for most N_{smooth}. At later times, at $t = 3$, we observed that C_4 performs better at $N_{smooth} = 64$ for GDSPH, the reason for which is not fully clear, and $N_{smooth} = 256$ for the ISPH method, where C_4 is the best-performing kernel of all at $t = 3$, likely due to the leading orders being much lower and thus giving more importance to higher-order errors. The noise of $N_{smooth} = 256$ for the ISPH method for C_4 is still greater than the $O_{I,256}$ kernel, as can be seen in Fig. 6. But the peak of the vortex is slightly higher with the C_4 kernel. We can compare the convergence rate (n_x^{-p}) to Cabezón & García-Senz (2024), who used ISPH and mix Sinc kernels to provide a more accurate and stable kernel against pairing instability. Those authors show a convergence rate of $p = 0.2$ at $N_{smooth} = 60$ to $p = 1.03$ at $N_{smooth} = 400$ after

$t = 1$. For ISPH/GDSPH($t = 3$)[$t = 1$] and the optimized kernels, we show a convergence rate of ($p = 0.5/p = 0.3$)[$p = 0.4/p = 0.3$] at $N_{smooth} = 32$, /($p = 0.6/p = 0.5$)[$p = 0.5/p = 0.5$] at $N_{smooth} = 64$, ($p = 0.9/p = 0.7$)[$p = 0.8/p = 0.7$] at $N_{smooth} = 128$ to ($p = 1.1/p = 0.9$)[$p = 1.0/p = 0.9$] at $N_{smooth} = 256$ after ($t = 3$)[$t = 1$]. In Wadsley et al. (2017), with modern artificial viscosity switches, they found a convergence rate of $p = 0.8$ for $N_{smooth} = 200$ with the C_4 kernel at $t = 3$. The moving mesh code AREPO in Springel (2010) yields a convergence rate of $p = 1.4$. Note that comparison to other modern SPH codes is slightly unfair, as we used a classic artificial viscosity scheme with constant α and β values, where we have chosen a low α value that gives a good result for the Gresho-Chan vortex.

For the Kelvin-Helmholtz instability, the biggest improvement could be seen for the GDSPH method, where the new optimized kernels captured similar results to that of the C_4 kernel at double N_{smooth} (both at $\Delta\rho = 2$ and $\Delta\rho = 8$ density contrast). For the ISPH method, the main improvement could be seen in angular momentum conservation of the low N_{smooth} runs, where $O_{I,32}$ performs better than C_2 and C_4. For the Sod-shocktube and Evrard collapse test, the main improvement of the optimized kernels could be seen as a reduction of the noise in the post-shock region.

In conclusion, we find that the linear-combination of kernels generates optimized kernels that significantly improve results, particularly in the low N_{smooth} regime. This can give a result similar to doubling the N_{smooth} of previous smoothing kernels (C_2, C_4) without any additional computational cost.

For ISPH, the results of the linear fluid behavior is only weakly dependent on the smoothing kernel and neighbor number (as seen in the Gresho-Chan vortex and the KH instability). We could see that the particle noise, angular momentum errors, and the mixing of the subsonic blob test improved with the new optimized kernels. For GDSPH, we observed an overall improvement with the optimized kernels for all test cases. From these tests we observed that ISPH give a significant improvement in convergence and behavior for subsonic flows compared to GDSPH. Results from the high Mach tests such as the Sod shocktube and Evrard collapse are much more insensitive to the SPH method and smoothing kernel used. For GDSPH, we recommend using O_{32} for $N_{smooth} <= 32$, O_{64} for $N_{smooth} <= 96$, and O_{256} for higher values of N_{smooth}. For ISPH, we recommend using $O_{I,32}$ for $N_{smooth} <= 32$, $O_{I,64}$ for $N_{smooth} <= 96$, and $O_{I,256}$ for higher values of N_{smooth}.

We have introduced a new IC generator to generate glass distributions of arbitrary density profiles in SPH. We have also introduced self-bias corrections to the kernel, that have been adjusted to the glass distribution instead of the lattice distribution of earlier work. This ensures more accurate density interpolation for the glass distribution of positive-definite kernels used in this paper.

This work shows that targeted kernel optimization can enhance SPH convergence beyond what is achievable with the standard compact kernels commonly used. It also demonstrates that the most effective kernel for a given problem depends on the specific SPH formulation used. These findings motivate future exploration of new optimization strategies and further improvements to both the smoothing kernels and the SPH method as a whole.

Data availability

The initial condition generator, optimized kernels, and test cases used in this work are publicly available at: https://github.com/robertwissing/testsuite.

Acknowledgements. We acknowledge support from the European High Performance Comput- ing Joint Undertaking (EuroHPC JU) and the Research Council of Norway through the funding of the SPACE Center of Excellence (grant agreement N0 101093441). Parts of the simulations were performed using the resources from the National Infrastructure for High Performance Computing and Data Storage in Norway, UNINETT Sigma2, allocated to Project NN9477K. We acknowledge the use of SPLASH (Price 2007) for the visualization of the results.

References

Agertz, O., Moore, B., Stadel, J., et al. 2007, MNRAS, 380, 963
Bonet, J. 1999, Comp. Methods Appl. Mech. Eng., 180, 97
Børve, S., Omang, M., & Trulsen, J. 2004, ApJS, 153, 447
Braspenning, J., Schaye, J., Borrow, J., & Schaller, M. 2023, MNRAS, 523, 1280
Brookshaw, L. 1985, PASA, 6, 207
Buhmann, M. D. 1998, Proc. Edinburgh Math. Soc., 41, 33
Buhmann, M. D. 2003, Radial Basis Functions (Cambridge: Cambridge University Press)
Cabezón, R. M., & García-Senz, D. 2024, MNRAS, 528, 3782
Cabezón, R. M., García-Senz, D., & Relaño, A. 2008, J. Comput. Phys., 227, 8523
Cabezón, R. M., García-Senz, D., & Figueira, J. 2017, A&A, 606, A78
Chernih, A., & Hubbert, S. 2014, J. Approx. Theory, 177, 17
Cullen, L., & Dehnen, W. 2010, MNRAS, 408, 669
Dehnen, W., & Aly, H. 2012, MNRAS, 425, 1068
Diehl, S., Rockefeller, G., Fryer, C. L., Riethmiller, D., & Statler, T. S. 2015, PASA, 32, e048
Evrard, A. E. 1988, MNRAS, 235, 911
Frontiere, N., Raskin, C. D., & Owen, J. M. 2017, J. Comput. Phys., 332, 160
García-Senz, D., Cabezón, R. M., & Escartín, J. A. 2012, A&A, 538, A9
García-Senz, D., Cabezón, R. M., Escartín, J. A., & Ebinger, K. 2014, A&A, 570, A14
Gresho, P. M., & Chan, S. T. 1990, Int. J. Numer. Methods Fluids, 11, 621
Lecoanet, D., McCourt, M., Quataert, E., et al. 2016, MNRAS, 455, 4274
Lombardi, J. C., Sills, A., Rasio, F. A., & Shapiro, S. L. 1999, J. Comput. Phys., 152, 687
Mandal, A., Mukherjee, D., & Mignone, A. 2023, ApJS, 268, 40
McNally, C. P., Lyra, W., & Passy, J.-C. 2012, ApJS, 201, 18
Menon, H., Wesolowski, L., Zheng, G., et al. 2015, Comput. Astrophys. Cosmol., 2, 1
Monaghan, J. J. 1988, Comp. Phys. Commun., 48, 89
Monaghan, J. J. 1992, ARA&A, 30, 543
Monaghan, J. J. 2005, Rep. Prog. Phys., 68, 1703
Morris, J. P. 1996, PASA, 13, 97
Price, D. J. 2007, PASA, 24, 159
Price, D. J. 2012, J. Comput. Phys., 231, 759
Price, D. J., & Monaghan, J. J. 2004, MNRAS, 348, 123
Price, D. J., Wurster, J., Tricco, T. S., et al. 2018, PASA, 35, e031
Read, J. I., & Hayfield, T. 2012, MNRAS, 422, 3037
Rosswog, S. 2009, New A Rev., 53, 78
Rosswog, S. 2015, MNRAS, 448, 3628
Rosswog, S. 2020, MNRAS, 498, 4230
Rosswog, S. 2025, Comp. Model. Eng. Sci., 143, 1713
Sandnes, T. D., Eke, V. R., Kegerreis, J. A., et al. 2025, J. Comput. Phys., 532, 113907
Schaback, R. 2011, Adv. Comput. Math., 34, 67
Shen, S., Wadsley, J., & Stinson, G. 2010, MNRAS, 407, 1581
Sod, G. A. 1978, J. Comput. Phys., 27, 1
Springel, V. 2010, MNRAS, 401, 791
Steinmetz, M., & Mueller, E. 1993, A&A, 268, 391
Thomas, P. A., & Couchman, H. M. P. 1992, MNRAS, 257, 11
Tricco, T. S. 2019, MNRAS, 488, 5210
Valdarnini, R. 2016, ApJ, 831, 103
Wadsley, J. W., Keller, B. W., & Quinn, T. R. 2017, MNRAS, 471, 2357
Wendland, H. 1995, Adv. Comput. Math., 4, 389
Wu, Z. 1995, Adv. Comput. Math., 4, 283

Appendix A: Differences between GDSPH and ISPH

In this appendix we include a resolution study between GDSPH and ISPH for the KH instability using the new optimized kernels for each respective N_{smooth}, which can be seen in Fig. A.1 for $\Delta\rho = 2$ and Fig. A.2 for $\Delta\rho = 8$. From these figures it is clear that the result for ISPH is highly independent of N_{smooth}, producing the same solution for $N_{smooth} = 32$ and $N_{smooth} = 128$. The GDSPH results are on the other hand, much more dependent on N_{smooth}, and exhibit severe damping of the instability at $N_{smooth} = 32$. At $N_{smooth} = 128$, the GDSPH method produces similar structures to that of the ISPH method at $\Delta\rho = 2$. For high density contrast $\Delta\rho = 8$, we can see that high N_{smooth} for the ISPH method generates more noise at the boundary layer at $t = 2$ compared to GDSPH in this case. We can also see that at later time, the high density structures are more dissipate in the ISPH method compared to the GDSPH method. We still see much more consistent behavior in the ISPH method compared to GDSPH method.

The high density contrast KH instability, is interesting as it is usually accompanied with strong surface tension effects in the Lagrangian formalism of SPH, due to high E_0 and E_1 errors at the discontinuous boundary. In GDSPH, these errors are reduced by doing a geometric density weighting in the gradient operator, reducing the errors at the density contrast. This sort of correction, deviates from the density estimate derivation of Lagrangian SPH and will thus introduce an entropy error. In GDSPH this is corrected for so that the method remains second-order accurate in entropy. ISPH removes the linear gradient error from it's gradients, which effectively removes more of the "partition" force that exist in SPH and only have E_0 error forces to remain ordered. This sort of correction also introduces entropy errors, which would potentially be first order in this case. In Fig. A.3, we show how one of the KH swirls develops when not using diffusion versus using diffusion for both the GDSPH and ISPH methods. Significant amounts of entropy bubbles can be seen in the ISPH version without diffusion, and a few can be seen in GDSPH. Adding thermal diffusion (Shen et al. 2010) allows for local mixing between the cold and hot phases, and we can see that we get good behavior in both ISPH and GDSPH. Slight numerical surface tension effects can be seen for GDSPH, while none can be seen for ISPH.

Appendix B: Kernel self-bias correction coefficients

In this section we present the self-correction bias coefficients present in

$$\rho_a = \rho_{a,estimate} - \epsilon m_a W(0, h_a) \tag{B.1}$$

$$\epsilon(N_{smooth}) = \frac{c_{i+1} - c_i}{N_{min,i+1} - N_{min,i}}(N_{smooth} - N_{min,i}) + c_i, \tag{B.2}$$

$$N_{\mathrm{min},i} \leq N_{\mathrm{smooth}} < N_{\mathrm{min},i+1},$$

where $N_{min,i} \in \{16, 24, 32, 48, 64, 96, 128, 256, 512\}$. The c_i coefficients can be found in Table B.1 for all kernels.

Appendix C: Kernel tables

The linear-combination of kernels after optimization are fitted are fitted to a polynomial in the form

$$P(x) = \sum_{i=0}^{8} c_i x^i, \tag{C.1}$$

$$W(q) = f_{norm}\left(1 + q^2 P(q)\right). \tag{C.2}$$

The polynomial coefficients for all the optimized kernels are given in C.1 and C.2.

Appendix D: Code implementation

A simple code snippet of the kernel function is included for clarification and for easier implementation, for both GDSPH and ISPH.

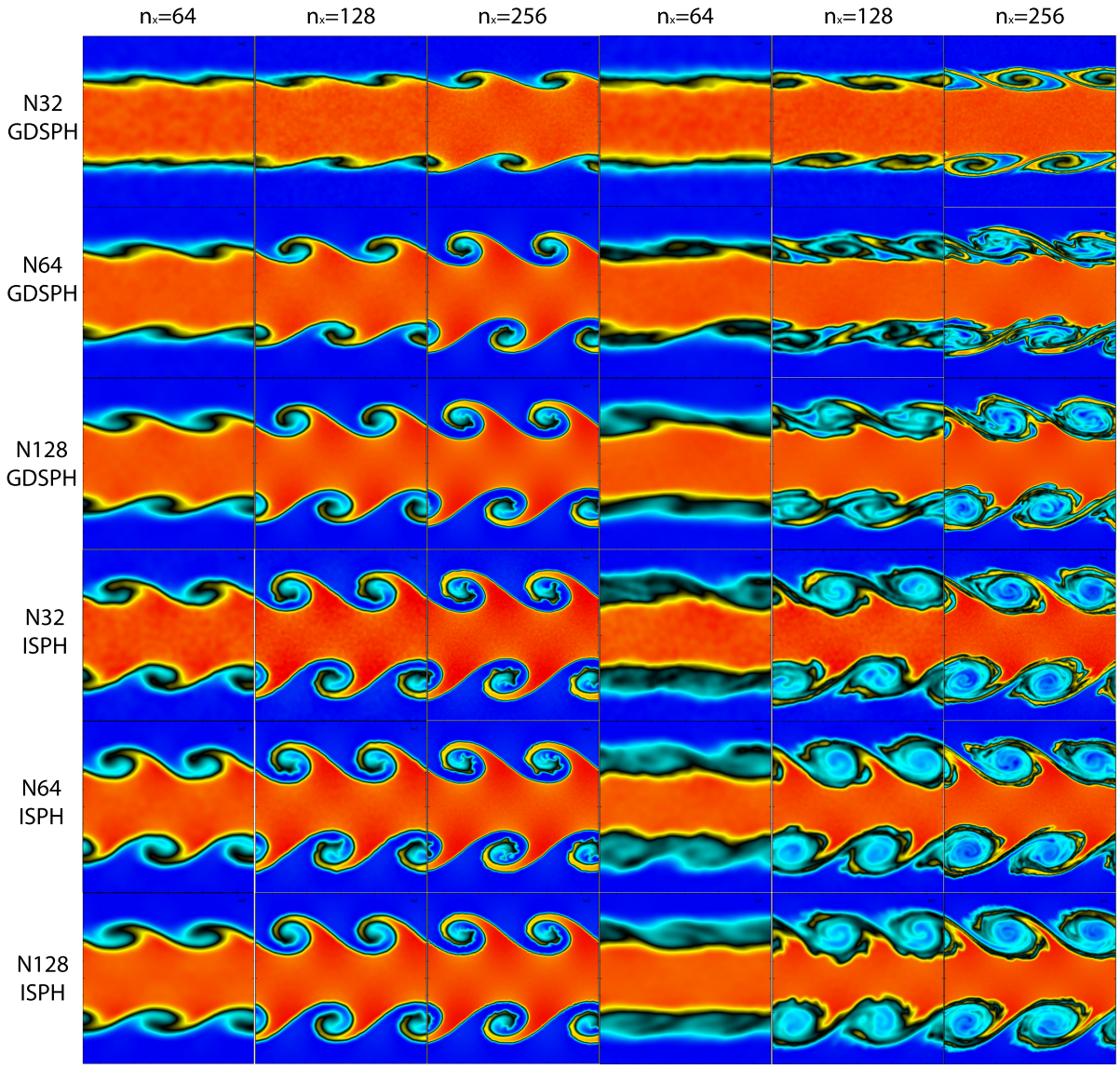

Fig. A.1. Density rendering of the Kelvin-Helmholtz instability ($\Delta\rho = 2$) at $t = 2$ (*three left columns*) and $t = 4$ (*three right columns*) for varying resolution ($n_x = 64, 128, 256$), $N_{smooth} = 32, 64, 128$, and method (GDSPH upper three rows and ISPH bottom three rows). The X and Y axis have a length of 1a. These simulations were all run with the optimized kernel for each $O_{N_{smooth}}$ GDSPH and $O_{I,N_{smooth}}$ for ISPH. We can see that ISPH manages to generate the same vortex structure, regardless of the number of neighbors if the resolution is high enough, while GDSPH struggles with lower number of neighbors even at the highest resolution.

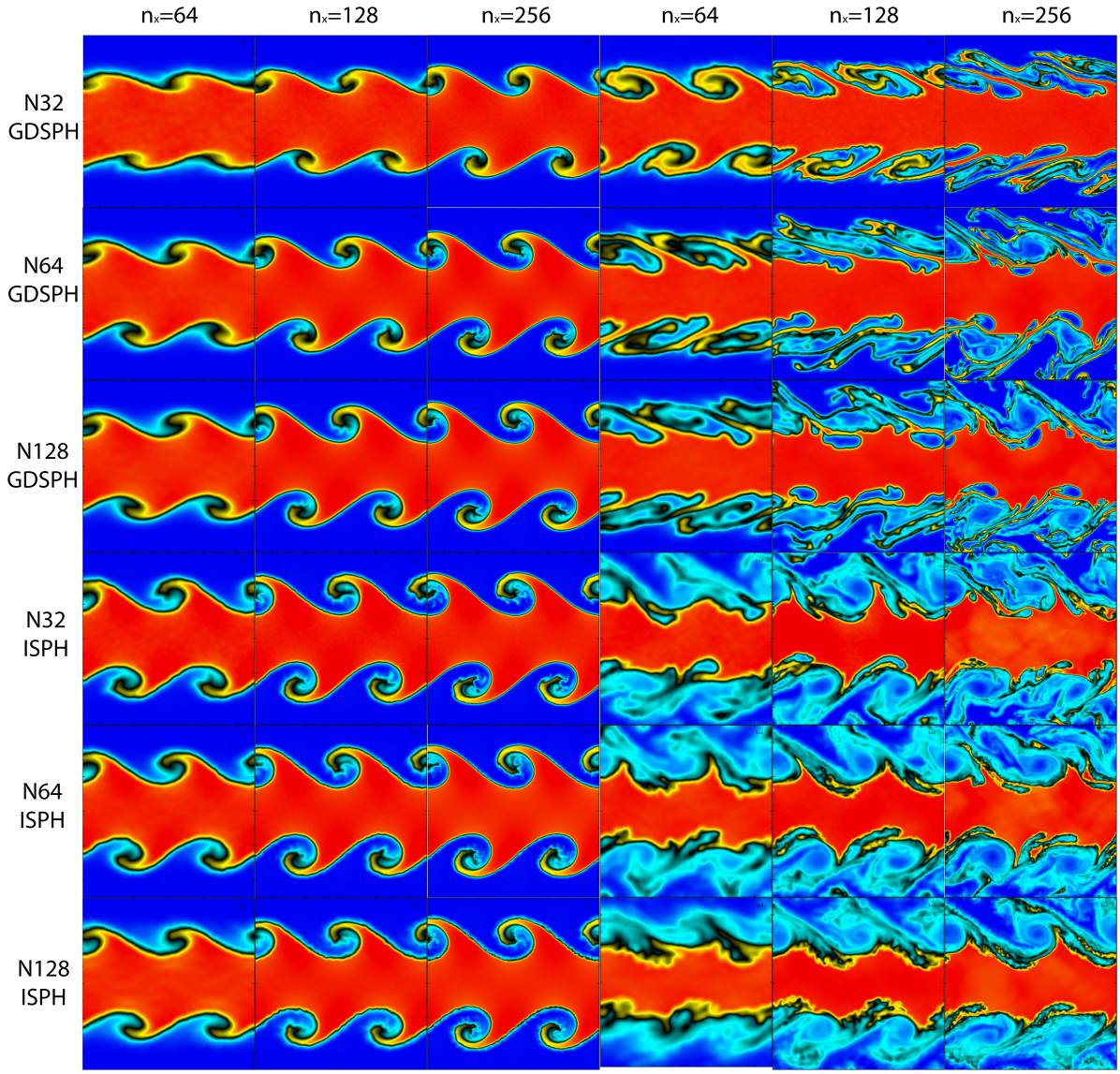

Fig. A.2. Density rendering of the Kelvin-Helmholtz instability ($\Delta\rho = 8$) at $t = 2$ (*three left columns*) and $t = 4$ (*three right columns*) for varying resolution ($n_x = 64, 128, 256$), $N_{smooth} = 32, 64, 128$, and method (GDSPH upper three rows and ISPH bottom three rows). The X and Y axis have a length of 1. These simulations were all run with the optimized kernel for each $O_{N_{smooth}}$ GDSPH and $O_{I,N_{smooth}}$ for ISPH. Similar as the low density version of Fig. A.1, we can see that ISPH captures instability well regardless of neighbor number. We can see additional diffusion from the ISPH runs compared to the GDSPH runs, even with the highest resolution. There also seem to be additional noise in the small-scale density structures as we increase the number of neighbors for ISPH. Potentially related to the issue found in Fig. A.3

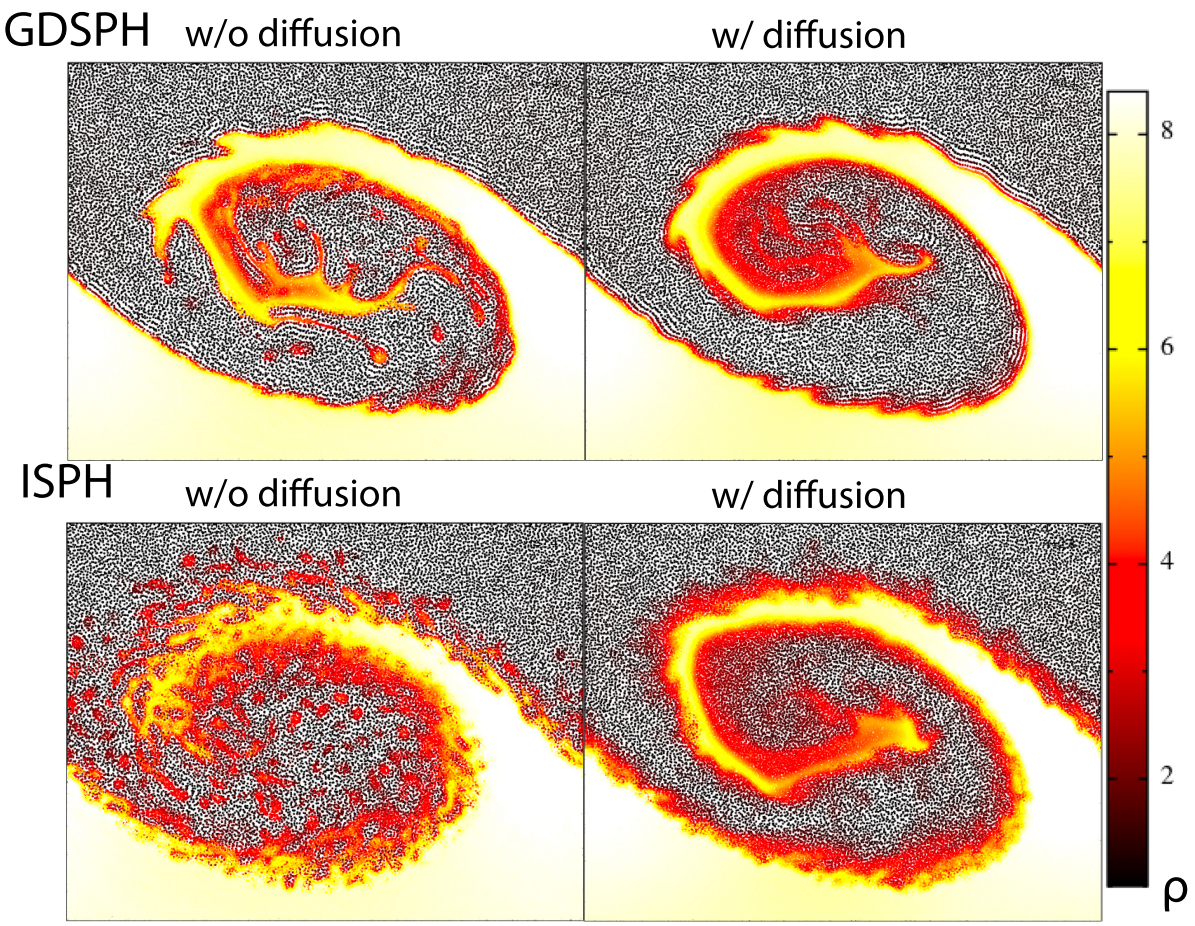

Fig. A.3. Particle density rendering of Kelvin-Helmholtz instability swirl at $t = 2$ for $\Delta\rho = 8$. The figures shows the different behavior of not using diffusion versus using diffusion for the GDSPH and ISPH methods. A significant number of entropy clumps can be seen being generated in the low density medium for ISPH, as the removal of linear gradient together with sharp boundaries causes chaotic noise at the boundary. Lagrangian SPH formalisms that are derived directly from the density estimate, provide full spatial conservation of entropy. However, deviation from the Lagrangian formalism, while still using the traditional density estimate means that we will introduce entropy errors in our solution. GDSPH corrects for first-order errors in entropy making it second-order in entropy. ISPH removes linear-gradient errors and will thus only be first-order accurate with entropy. The issue might be further aggravated by the use of average gradient kernels within ISPH. Adding thermal diffusion allows for local mixing between the cold and hot phase, and we can see that we get good behavior in both ISPH and GDSPH. Slight numerical surface tension effect can be seen for GDSPH, while none can be seen for ISPH.

Table B.1. Self-bias correction for GDSPH. The $c_{N_{smooth}}$ is to be used in Eq. B.2 to calculate the correction factor for density Eq. B.1.

	c_1	c_2	c_3	c_4	c_5	c_6	c_7	c_8	c_9
O_{32}	0.7765625	0.85302734	0.86884766	0.89765625	0.92294922	0.93652344	0.94716797	0.96826172	0.98261719
O_{64}	0.83125305	0.93046875	0.93710937	0.94462891	0.95146484	0.96054687	0.96875	0.9828125	0.99072266
O_{128}	0.83544617	0.95097656	0.95742187	0.96210937	0.96582031	0.97236328	0.97832031	0.98867188	0.99404297
O_{256}	0.84145508	0.95	0.97675781	0.98193359	0.98554688	0.99023438	0.99355469	0.99814453	0.99931641
$O_{I,32}$	0.67424316	0.85517578	0.88662109	0.88674219	0.92255859	0.95058594	0.97558594	1.0	1.0
$O_{I,64}$	0.74882812	0.85976562	0.87744141	0.90087891	0.91757812	0.93232422	0.94550781	0.96835938	0.9828125
$O_{I,128}$	0.80136719	0.90957031	0.92109375	0.93486328	0.94765625	0.96025391	0.97089844	0.98574219	0.99296875
$O_{I,256}$	0.81279297	0.91591797	0.92558594	0.93730469	0.94824219	0.95908203	0.96845703	0.9828125	0.99091797
C_2	0.80507812	0.90039062	0.91347656	0.93349609	0.95693359	0.97158203	0.97675781	0.98847656	0.99394531
C_4	0.70195312	0.87412109	0.91611328	0.95019531	0.96748047	0.98417969	0.99013672	0.99658203	0.99873047
C_6	0.54921875	0.76171875	0.83505859	0.92001953	0.96542969	0.98515625	0.99130859	0.99814453	0.99941406
C_8	0.43124695	0.63945312	0.72568359	0.85019531	0.93779297	0.98095703	0.99003906	0.9984375	0.99960938
WU_2	0.94551392	0.97011719	0.98007812	0.98652344	0.98964844	0.99326172	0.99482422	0.99707031	0.99824219
WU_4	0.78925781	0.96679688	0.98798828	0.99375	0.99619141	0.99775391	0.99863281	0.99941406	0.99960938
CM_{04}	0.8734375	0.89990234	0.91533203	0.94384766	0.95839844	0.97167969	0.97919922	0.99277344	1.0
CM_{05}	0.74023437	0.80107422	0.82080078	0.85361328	0.8921875	0.91513672	0.92734375	0.95458984	0.97119141
BUH	0.7812439	0.86464844	0.88203125	0.90712891	0.93808594	0.95830078	0.96572266	0.98271484	0.99082031

Notes. We can see that all kernels converge toward 1 at higher N_{smooth}.

Table B.2. Multipliers to calculate self-bias correction for ISPH from the values of Table B.1.

	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9
O_{32}	1.007	1.003	1.003	1.0	0.999	0.999	0.996	0.996	0.995
O_{64}	1.005	0.999	1.0	0.999	0.999	0.998	0.996	0.997	0.995
O_{128}	1.004	0.998	0.999	0.999	0.998	0.996	0.995	0.997	0.996
O_{256}	1.004	1.0	0.999	0.999	0.998	0.998	0.998	0.998	0.998
$O_{I,32}$	1.019	0.979	0.992	0.970	0.886	0.803	0.714	0.341	0.088
$O_{I,64}$	1.007	1.001	1.002	1.001	1.0	0.999	0.996	0.997	0.996
$O_{I,128}$	1.005	1.0	1.0	1.0	1.0	1.041	0.996	0.997	0.996
$O_{I,256}$	1.005	1.0	1.0	1.0	1.0	0.999	0.997	0.998	0.996
C_2	1.005	0.996	0.995	0.994	0.988	0.998	0.998	0.996	0.997
C_4	0.963	0.997	0.997	0.999	0.999	0.998	1.0	0.999	0.999
C_6	0.973	0.997	0.998	1.0	1.0	1.0	1.0	1.0	1.0
C_8	0.974	0.998	0.999	0.999	1.0	1.0	1.0	1.0	1.0
WU_2	0.978	0.990	0.987	0.956	0.937	0.895	0.853	0.693	0.363
WU_4	1.003	0.998	0.998	0.997	0.997	0.993	0.994	0.995	0.993
CM_{04}	0.981	0.975	0.910	0.806	0.719	0.557	0.395	0.1	x
CM_{05}	0.976	1.0	1.0	1.0	0.998	0.999	0.997	0.998	0.997
BUH	0.970	0.996	0.997	0.998	0.994	0.999	0.998	0.998	0.998

Notes. Correction to self-bias for ISPH is given by using $(f_i \cdot c_i)$ instead of c_i in Eq. B.2. In general the kernels with ISPH have very similar self-bias coefficients to that of GDSPH, but one can see that some kernels in ISPH become increasingly unstable as N_{smooth} increases, such as CM_{04}, WU_2, and $O_{I,32}$, and their behavior diverges from the others, which otherwise converges to roughly 1.

Table C.1. Polynomial coefficients for all the optimized kernels based on the hydrostatic glass test case.

	Norm	α_0	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8
$O_{glass,32}$	3.0336	-12.7824	56.1920	-182.1975	398.3558	-530.5896	413.1597	-173.6614	30.5234	0.0
$O_{glass,64}$	3.0599	-7.1464	8.4866	-4.6008	22.5183	-44.3100	26.6288	6.6825	-13.2333	3.9743
$O_{glass,128}$	2.9693	-7.104	6.2483	12.4432	-28.5090	29.4627	-22.3099	11.2888	-2.5203	0.0
$O_{glass,256}$	3.0302	-7.5780	10.7478	-12.9728	55.6311	-132.8491	162.5874	-110.9071	40.5189	-6.1782

Table C.2. Polynomial coefficients for all the optimized kernels based on the Gresho-Chan test case.

	Norm	α_0	α_1	α_2	α_3	α_4	α_5	α_6	α_7	α_8
O_{32}	3.0702	-15.1790	73.5187	-257.6388	645.4938	-1078.5898	1171.7511	-797.9519	310.1745	-52.5786
O_{64}	3.1160	-10.7948	34.9240	-85.6680	148.6523	-124.7885	-11.0621	104.2637	-73.5505	17.0239
O_{128}	3.2790	-9.2841	21.1601	-34.5440	40.3544	24.1265	-149.8027	189.8576	-105.1688	22.3011
O_{256}	3.3158	-6.9471	0.6117	40.6052	-109.3938	199.6857	-269.1855	230.9765	-108.5062	21.1535
$O_{I,32}$	3.4003	-37.1051	286.5916	-1109.5005	2409.0689	-3009.6819	2067.8803	-618.2116	-36.1322	46.0904
$O_{I,64}$	3.4319	-15.2178	66.9679	-205.6619	448.2313	-637.1402	569.1468	-307.2116	91.3253	-11.4399
$O_{I,128}$	3.2796	-10.3851	23.4816	-17.8825	-44.4497	195.2885	-338.6360	309.5800	-146.1530	28.1562
$O_{I,256}$	3.1813	-10.9758	34.0224	-79.6218	145.2028	-156.0396	69.5784	17.6550	-28.5200	7.6987

Table C.3. Contribution of each kernel in the linear combination used to optimize all kernels for the Gresho-Chan test case.

	$O_{g,64}$	$O_{g,128}$	$O_{g,256}$	CM_{04}	WU_2	BUH	CM_{05}	C2	C4	C6	C8	WU_4
O_{32}	-1.630	0.604	0.291	0.223	0.030	0.138	0.289	-0.173	-0.080	0.046	-0.003	0.016
O_{64}	-0.032	0.155	0.000	0.000	-0.039	0.045	-0.140	0.155	0.097	-0.017	-0.003	-0.047
O_{128}	0.424	-0.158	-0.068	-0.011	-0.113	0.034	-0.215	0.278	0.129	-0.035	-0.003	0.071
O_{256}	0.603	-0.087	0.227	0.075	-0.058	-0.189	-0.355	0.371	0.102	-0.023	-0.008	0.261
$O_{I,32}$	-1.397	0.530	0.023	0.008	0.018	-0.171	0.099	-0.073	-0.023	0.007	0.000	0.009
$O_{I,64}$	-1.397	0.530	-0.088	-0.005	-0.014	0.018	0.047	-0.027	0.094	-0.089	0.034	-0.027
$O_{I,128}$	-0.670	0.377	0.000	0.000	0.051	-0.009	0.085	-0.136	0.015	-0.013	0.004	-0.020
$O_{I,256}$	-1.131	0.536	-0.012	-0.005	0.046	0.118	0.012	-0.152	-0.027	0.006	0.006	0.025

```
/** ISPH VERSION
 * @brief kernelOptimized is an optimized kernel for SPH, with different form for different nsmooth
 *
 * This kernel is explained and defined in Wissing (2025).  The kernel
 * is a linear combination of several positive definite kernels,
 * which coeffecients have been optimized to give the best result for the gresho chan vortex
 * r = |dx|/H
 * And for us, H = 2*h_smooth
 *
 * Includes a correction for self-interactions.
 * Which give density of 1 for a glass relaxed with the kernel.
 *
 * NOTE: this kernel should not be called for r > 1
 * @param ar2 = (|dx|/h)^2 = (2r)^2
 * @return (pi h^3) W
 */
inline double kernelOptimized(double ar2, int nSmooth, double mnorm)
{
  double ak;
  double Rkern2=4.0;
  double Wzero=1.0,norm;
  double c0,c1,c2,c3,c4,c5,c6,c7,c8;
  //Wzero values for piecewise function
  double n16,n24,n32,n48,n64,n96,n128,n192,n256,n512;
  if (nSmooth <= 32) //O_I,32 Wissing et al. 2025
    {
      n16=0.68671868118; n24=0.8375001518; n32=0.87988276971;
      if(nSmooth<=16) Wzero=(n16-0.40)/(14)*(nSmooth-2)+0.40;
      if(nSmooth<=24) Wzero=(n24-n16)/8*(nSmooth-16)+n16;
      if(nSmooth<=32) Wzero=(n32-n24)/8*(nSmooth-24)+n24;
      norm=(3.4003319462332073/8./M_1_PI);
      c0=-37.10514490418083;
      c1=286.59163136982374;
      c2=-1109.5005318509457;
      c3=2409.0689549866306;
      c4=-3009.6818640918887;
      c5=2067.880326832294;
      c6=-618.2115694970241;
      c7=-36.132229318136154;
      c8=46.09042647341556;
    }
  else if (nSmooth <= 96) //O_I,64 Wissing et al. 2025
    {
      n32=0.87900413315; n48=0.90136718636; n64=0.91787082742; n96=0.9315429323;
      if(nSmooth<=48) Wzero=(n48-n32)/16*(nSmooth-32)+n32;
      if(nSmooth<=64) Wzero = (n64-n48)/16*(nSmooth-48)+n48;
      if(nSmooth<=96) Wzero = (n96-n64)/32*(nSmooth-64)+n64;
      norm=(3.4318494020523547/8./M_1_PI);
      c0=-15.217780383519472;
      c1=66.96785864668652;
      c2=-205.6618857567194;
      c3=448.23132682213446;
      c4=-637.1401597525037;
      c5=569.1467883955034;
      c6=-307.2115693839176;
      c7=91.32532556233177;
      c8=-11.439904149996446;
    }
  else //O_I,256 Wissing et al. 2025
    {
      n96=0.95820351086; n128=0.96523439238; n192=0.9764745855; n256=0.98056677343; n512=0.98710987224;
      if(nSmooth<=128) Wzero = (n128-n96)/32*(nSmooth-96)+n96;
      if(nSmooth<=192) Wzero = (n192-n128)/64*(nSmooth-128)+n128;
      if(nSmooth<=256) Wzero = (n256-n192)/64*(nSmooth-192)+n192;
      if(nSmooth<=512) Wzero = (n512-n256)/256*(nSmooth-256)+n256;
      norm=(3.1813352911279 15/8./M_1_PI);
      c0=-10.975827966498654;
      c1=34.022346938397256;
      c2=-79.62182205737668;
      c3=145.2028297298656;
      c4=-156.0395550888878;
      c5=69.57842722118721;
      c6=17.65497978738222;
      c7=-28.52004202463561;
      c8=7.698663460565884;
    }

  if (ar2 <= 0) ak = norm*Wzero;
  else if (ar2 >= Rkern2) ak = 0.0;
  else {
    double au = sqrt(ar2*0.25);
    ak = norm*(1+au*(au*(c0+au*(c1+au*(c2+au*(c3+au*(c4+au*(c5+au*(c6+au*(c7+c8*au))))))))));
//  Gradient adk=(pi h^5/|dx|^2) (dx.dot.gradW)
//  adk = (norm/4.)*((c0*2+au*(c1*3+au*(c2*4+au*(c3*5+au*(c4*6+au*(c5*7+au*(c6*8+au*(c7*9+c8*10*au))))))))));
  }
  return ak;
}
```

```
/** GDSPH VERSION
 * @brief kernelOptimized is an optimized kernel for SPH, with different form for different nsmooth
 *
 * This kernel is explained and defined in Wissing (2025).  The kernel
 * is a linear combination of several positive definite kernels,
 * which coeffecients have been optimized to give the best result for the Gresho-Chan vortex
 * r = |dx|/H
 * And for us, H = 2*h_smooth
 *
 * Includes a correction for self-interactions.
 * Which give density of 1 for a glass relaxed with the kernel.
 *
 * NOTE: this kernel should not be called for r > 1
 * @param ar2 = (|dx|/h)^2 = (2r)^2
 * @return (pi h^3) W
 */
inline double kernelOptimized(double ar2, int nSmooth, double mnorm)
{
  double ak;
  double Wzero=1.0,norm;
  double Rkern2=4.0;
  double c0,c1,c2,c3,c4,c5,c6,c7,c8;
  //Wzero values for piecewise function
  double n16,n24,n32,n48,n64,n96,n128,n192,n256,n512;
  if (nSmooth <= 32) //O_32 Wissing et al. 2025
    {
      n16=0.7765625; n24=0.85302734; n32=0.86884766;
      if(nSmooth<=16) Wzero=(n16-0.40)/(14)*(nSmooth-2)+0.40;
      if(nSmooth<=24) Wzero=(n24-n16)/8*(nSmooth-16)+n16;
      if(nSmooth<=32) Wzero=(n32-n24)/8*(nSmooth-24)+n24;
      norm=(3.07023991432113334/8./M_1_PI);
      c0=-15.178969768800927;
      c1=73.51866968356427;
      c2=-257.63874599152774;
      c3=645.4937717480914;
      c4=-1078.589778178365;
      c5=1171.7510466338265;
      c6=-797.951908267712;
      c7=310.1744769215709;
      c8=-52.578562780647644;
    }
  else if (nSmooth <= 96) //O_64 Wissing et al. 2025
    {
      n32=0.93710937; n48=0.94462891; n64=0.95146484; n96=0.96054687;
      if(nSmooth<=48) Wzero=(n48-n32)/16*(nSmooth-32)+n32;
      if(nSmooth<=64) Wzero = (n64-n48)/16*(nSmooth-48)+n48;
      if(nSmooth<=96) Wzero = (n96-n64)/32*(nSmooth-64)+n64;
      norm=(3.1160348893350283/8./M_1_PI);
      c0=-10.794751909300937;
      c1=34.923977443679455;
      c2=-85.66799299517906;
      c3=148.65227923861258;
      c4=-124.78847965696686;
      c5=-11.062047962891707;
      c6=104.26365151520089;
      c7=-73.55050556719418;
      c8=17.023869894040175;
    }
      else //O_256 Wissing et al. 2025
    {
      n96=0.99023438; n128=0.99355469; n192=0.99746094; n256=0.99814453; n512=0.99931641;
      if(nSmooth<=128) Wzero = (n128-n96)/32*(nSmooth-96)+n96;
      if(nSmooth<=192) Wzero = (n192-n128)/64*(nSmooth-128)+n128;
      if(nSmooth<=256) Wzero = (n256-n192)/64*(nSmooth-192)+n192;
      if(nSmooth<=512) Wzero = (n512-n256)/256*(nSmooth-256)+n256;
      norm=(3.3158322817560615/8./M_1_PI);
      c0=-6.94709887927258;
      c1=0.6116847826437315;
      c2=40.605173332308;
      c3=-109.393837050743O6;
      c4=199.68569324945807;
      c5=-269.18551329586074;
      c6=230.97653909776236;
      c7=-108.50616849023058;
      c8=21.153527253935298;
    }

  if (ar2 <= 0) ak = norm*Wzero;
  else if (ar2 >= Rkern2) ak = 0.0;
  else {
    double au = sqrt(ar2*0.25);
    ak = norm*(1+au*(au*(c0+au*(c1+au*(c2+au*(c3+au*(c4+au*(c5+au*(c6+au*(c7+c8*au))))))))));
//  Gradient adk=(pi h^5/|dx|^2) (dx.dot.gradW)
//  adk = (norm/4.)*((c0*2+au*(c1*3+au*(c2*4+au*(c3*5+au*(c4*6+au*(c5*7+au*(c6*8+au*(c7*9+c8*10*au))))))))));
  }
  return ak;
}
```